UNIVERSIDADE Ð COIMBRA

Andreia Próspero Guerreiro

Portfolio Selection in Evolutionary Algorithms

Doctoral thesis submitted to the Doctoral Program in Information Science and Technology, supervised by Professor Carlos Manuel Mira da Fonseca, and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2018

Portfolio Selection in Evolutionary Algorithms

A thesis submitted to the University of Coimbra
in partial fulfillment of the requirements for the
Doctoral Program in Information Science and Technology

by

Andreia Próspero Guerreiro

apg@dei.uc.pt

Department of Informatics Engineering
Faculty of Sciences and Technology
University of Coimbra

September 2018

Financial support by
Fundação para a Ciência e a Tecnologia
Ref.: SFRH/BD/77725/2011
Portfolio Selection in Evolutionary Algorithms
©2018 Andreia P. Guerreiro

Advisor

Prof. Carlos Manuel Mira da Fonseca

Associate Professor
Department of Informatics Engineering
Faculty of Sciences and Technology of University of Coimbra

Dedicated to my parents, sister, and Daniel

Abstract

Evolutionary Algorithms (EAs) are algorithms inspired in the process of natural selection, and are among the most popular methods in multiobjective optimization. As objectives are typically conflicting with one another, instead of a single optimal solution, there is usually a set of optimal solutions which, together, form a trade-off surface. The choice of the best solution depends on the preferences of a Decision Maker (DM), which arise from subjective information not contained in the multiobjective optimization problem formulation itself. In the absence of such preference information, an optimization algorithm should be able to cover the optimal trade-off surface as well as possible, in order to increase the chance that at least one satisfactory solution is presented to the (unknown) DM.

One of the main steps of an EA is selection which is aimed at focusing on the best solutions, but must also maintain a sufficient level of diversity in the population. A very successful selection approach consists in optimizing a measure of the quality of the population as a whole, accounting simultaneously for individual quality and population diversity. Such EAs are called indicator-based EAs, as they rely on set-quality indicators to perform selection. Indicator-based EAs are currently state-of-the-art algorithms in Evolutionary Multiobjective Optimization (EMO).

More recently, the notion of selection in EAs has been linked to the Portfolio Selection Problem (PSP), which is well known in Finance. In this analogy, individuals are seen as assets whose returns are random variables that are characterized by their expected values and covariance matrix. Balancing between good and diverse solutions in a population becomes analogous to balancing expected return and risk, respectively, in financial portfolios. In particular, it has been empirically shown that such a balance in EMO selection can be achieved using the risk-adjusted performance index known as Sharpe ratio, without modification, as a new quality indicator in the context of a particular formulation of random individual return related to the concept of dominated hypervolume.

The focus of this thesis is the subset selection problem at the core of selection in EMO algorithms, but from the more general point of view of Portfolio Selection. Two lines of work are followed. The first one consists in improving those state-of-the-art algorithms that are based on the hypervolume indicator, which is a well known and theoretically supported, but computationally expensive, quality indicator. The second line of work goes beyond subset selection: a new type of indicators based on the Sharpe ratio is studied, both theoretically and experimentally, shedding new light on selection, fitness assignment and preference integration in EMO algorithms.

Keywords: Evolutionary Algorithms, Multiobjective Optimization, Portfolio Selection, Subset Selection, Hypervolume Indicator, Sharpe Ratio

Resumo

Os Algoritmos Evolutivos (AEs) são algoritmos inspirados no processo de selecção natural e estão entre os métodos mais utilizados em optimização multiobjectivo. Como tipicamente os objectivos são contraditórios entre si, em vez de existir uma única solução óptima, existe um conjunto de soluções óptimas que, juntas, formam uma superfície de trade-off. A escolha da melhor solução depende das preferências do decisor, que resultam de informação subjectiva não contida na formulação do problema multiobjectivo. Na ausência desta informação sobre preferências, um algoritmo de optimização deve ser capaz de cobrir a superfície de trade-off o melhor possível, de modo a aumentar as hipóteses de que, pelo menos, uma solução satisfatória seja apresentada ao decisor (desconhecido).

Um dos principais passos de um AE é a selecção, que visa priveligiar as melhores soluções mas deve também manter um nível suficiente de diversidade na população. Uma abordagem bem sucedida para selecção consiste em optimizar uma medida de qualidade da população como um todo, tendo simultaneamente em conta a qualidade de cada indivíduo e a diversidade da população. A estes AEs dá-se o nome de AEs baseados em indicadores, e estão actualmente entre o estado da arte em Optimização Multiobjectivo Evolutiva (OME).

Recentemente, a noção de selecção em AEs tem sido ligada ao Problema de Selecção de Portfólios (PSP), que é muito conhecido em Finanças. Nesta analogia, os indivíduos são vistos como activos cujos retornos são variáveis aleatórias caracterizadas pelo seus valores esperados e por uma matriz de covariância. Balancear entre boas soluções e soluções diferentes numa população torna-se análogo a balancear retorno esperado e risco, respectivamente, em portfólios financeiros. Em particular, já foi mostrado empiricamente que este balanceamento na selecção em OME pode ser obtido usando, sem qualquer modificação, um índice de desempenho que tem em conta o risco e que é conhecido como rácio de Sharpe, como um novo indicador de qualidade no contexto de uma determinada formulação de retorno individual aleatório relacionado com o conceito de hipervolume dominado.

O foco desta tese é o problema de selecção de subconjuntos que está no cerne da selecção em algoritmos de OME, mas do ponto-de-vista mais geral de Selecção de Portfólios. Foram seguidas duas linhas de trabalho. A primeira consiste em melhorar os algoritmos estado da arte baseados no indicador de hipervolume, que é um indicador muito conhecido e com bases teóricas, mas cujo cálculo é computacionalmente dispendioso. A segunda linha de trabalho vai além da selecção de subconjuntos: estudar um novo tipo de indicadores baseados no rácio de Sharpe, tanto teórica como experimentalmente, abrindo uma nova perspectiva sobre a selecção, atribuição de aptidão e integração de preferências em algoritmos de OME.

Palavras-chave: Algoritmos Evolutivos, Optimização Multiobjectivo, Selecção de Portfólios, Selecção de Subconjuntos, Indicador de Hipervolume, Rácio de Sharpe

Acknowledgements

First of all, I am thankful to my parents for the education they gave me, for allowing and helping me pursue an academic degree and end up doing one of the things I like the most, research. I owe a big thank you to you and my sister for all the support during the PhD and for all the patience, specially for putting up with me in that period that I forbid you to ask me how was the thesis going! I'm deeply grateful to Daniel, for being there, for making me laugh, and for all the support, specially in this last year. Thank you for helping me go through all of my frustrations, for all the incentives when I felt like this PhD was never going to end and felt like turning my back on this.

I want to thank all my PhD friends, those whom I shared the ECOS lab with (Pedro, Maryam, Nuno and Jean), and the others that also accompanied me in this journey (Filipe and Alcides). Thank you all for the friendship, for all the good times we shared, for all the funny moments, and for being an inspiration to me. I also have to thank all the friends that I had the pleasure to share the D. Sancho house with (Vera, Margarida, Marina and Danielas). Thank you for all the good moments, the dinners, the wii parties, the talking, the pranks, the laughing! Thank you for all the evenings we shared, only when I left D. Sancho I realized how important these were in my journey.

I am thankful to my advisor, professor Carlos Fonseca, for encouraging me into research so many years ago when I was only an undergraduate student, and for mentoring me since then. Thank you for proposing the topic of my PhD, and for pushing me a bit out of my scientific comfort zone. I have learned a lot from you, thank you for that and for the availability to discuss my ideas and my doubts, for all the help, and for encouraging me. I am also thankful to professor Luís Paquete, who I think was almost like a co-advisor. Thank you for the collaborations, for all the help, for the things I learned from you, and for sometimes challenging me with interesting research questions. I also want to thank professor Ricardo Takahashi for receiving and mentoring me for 3 months in the Federal University of Minas Gerais, in Brazil, and which originated the work in Section 5.6.

Finally, to all of those that I mentioned and other people, the long-date friends, the people that I met along this journey, even if they were part of it only briefly, I owe you a big thank you, you all contributed to make this a better journey, and to get me to the end of this PhD with (some) sanity!

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Summary . . . 2
1.3 Contributions . . . 3

2 Multiobjective Optimization **5**
2.1 Optimization . . . 5
2.1.1 Single-Objective Case . . . 6
2.1.2 Multiobjective Case . . . 7
2.2 Decision Making . . . 8
2.2.1 Decision Analysis . . . 9
2.2.2 Evolutionary Multiobjective Optimization . . . 10
2.3 Evolutionary Algorithms . . . 11
2.3.1 Algorithm . . . 11
2.3.2 Evolutionary Multiobjective Optimization Algorithms . . . 14
2.4 Selection in EMOAs . . . 16
2.4.1 Methods Focused on Individual Quality . . . 17
2.4.2 Diversity Preservation Techniques . . . 18
2.4.3 Decomposition Methods . . . 20
2.4.4 Methods Based on Set Quality . . . 20
2.4.5 Portfolio Selection Approach . . . 22
2.4.6 Properties of Selection Methods . . . 27
2.5 Concluding Remarks . . . 31

3 Hypervolume Indicator **33**
3.1 Hypervolume-related Problems . . . 34
3.1.1 Notation . . . 34
3.1.2 Definitions . . . 34
3.1.3 Problems . . . 38
3.2 Properties . . . 41
3.3 Hypervolume-based Archiving Algorithms . . . 42
3.3.1 Classes . . . 42
3.3.2 Convergence Analysis . . . 43
3.4 Algorithms . . . 45
3.4.1 Techniques . . . 46
3.4.2 Hypervolume Indicator . . . 51
3.4.3 Hypervolume Contributions . . . 59
3.4.4 Exact Algorithms for the HSSP . . . 65

3.4.5 Greedy Algorithms for the HSSP . . . 70
3.5 Data Sets . . . 75
3.6 Concluding Remarks . . . 77

4 Hypervolume Subset Selection 79
4.1 Computation of the Hypervolume Indicator . . . 80
4.1.1 Three Dimensions . . . 80
4.1.2 Four Dimensions . . . 86
4.1.3 Experimental Results . . . 86
4.1.4 Practical Implications . . . 88
4.2 Computation of All Hypervolume Contributions . . . 89
4.2.1 Three Dimensions . . . 89
4.2.2 Four Dimensions . . . 95
4.2.3 Experimental Results . . . 95
4.2.4 Practical Implications . . . 98
4.3 Decremental Greedy Algorithms for the HSSP . . . 98
4.3.1 General Algorithm Revisited . . . 98
4.3.2 2-Dimensional Example . . . 99
4.3.3 Algorithms for $d = 3, 4$ . . . 100
4.3.4 A Naive Bound for the Approximation Ratio to the HSSP . . . 100
4.4 Incremental Greedy Algorithms for the HSSP . . . 101
4.4.1 General Algorithm Revisited . . . 102
4.4.2 2-Dimensional Example . . . 102
4.4.3 Algorithm for $d = 2$ . . . 104
4.4.4 Computing OneContribution in $d = 3$ . . . 104
4.4.5 Algorithm for $d = 3$ . . . 106
4.4.6 No Approximation Ratio for the HSSPComplement . . . 116
4.5 Comparison of Greedy Algorithms for the HSSP . . . 117
4.5.1 Approximation Quality . . . 118
4.5.2 Selected Subsets . . . 122
4.5.3 Runtime . . . 130
4.5.4 Remarks on the Greedy Algorithms . . . 132
4.6 Greedy Hypervolume-based Archiving Algorithms . . . 133
4.6.1 Convergence Quality . . . 134
4.7 Concluding Remarks . . . 141

5 Portfolio Selection 143
5.1 Notation . . . 144
5.2 Sharpe-Ratio Indicator . . . 144
5.3 Hypervolume Sharpe-Ratio Indicator (HSR) . . . 144
5.3.1 Mathematical Analysis of the HSR Indicator . . . 145
5.4 Properties of the HSR Indicator . . . 146
5.4.1 Reference Points and Linear Scaling . . . 146
5.4.2 Monotonicity . . . 148
5.4.3 Optimal μ-Distributions on Two-Objective Linear Fronts . . . 151
5.4.4 Comparison with the Hypervolume Indicator . . . 160
5.5 Experimental Results . . . 163
5.5.1 Optimal Investment . . . 163

5.5.2 Approximating Optimal μ-Distributions 167
5.6 Extended Hypervolume Sharpe-Ratio Indicator 169
5.6.1 Motivation 172
5.6.2 Formulation 177
5.6.3 Experiments and Discussion 179
5.7 Preferability Hypervolume Sharpe-Ratio Indicator 181
5.7.1 Formulation 182
5.8 Discussion and Concluding Remarks 184

6 Portfolio Selection in EMOAs: Experimental Results 187
6.1 EMO Algorithms 188
6.1.1 Implementation Details 188
6.2 Problem Description 190
6.2.1 DTLZ Fronts 190
6.2.2 Multiobjective Knapsack Problem 191
6.3 Experimental Results 193
6.3.1 Parameter Settings 193
6.3.2 Results 194
6.3.3 Hypervolume-Based EMOAs 204
6.3.4 Sharpe-Ratio-Based EMOA 206
6.4 Concluding Remarks 210

7 Conclusion 211
7.1 Hypervolume-Based Selection 212
7.2 Portfolio-Based Selection 212
7.3 Future Work 213
7.4 Final Remarks 214

List of Figures

2.1 Decision Maker and Evolutionary Algorithms interaction. 10
2.2 Evolutionary Algorithm. 12
2.3 A crossover example. 13
2.4 Diversity preservation techniques. 19
2.5 Hypervolume Indicator. 21
2.6 Examples of the regions measured to compute p_i and p_{ij} 25
2.7 The optimal investment 26

3.1 Hypervolume-related definitions: 2D examples 36
3.2 Hypervolume related definitions: 3D examples 36
3.3 Joint Contribution: 3D example 37
3.4 Examples of delimiters in the presence of repeated coordinates 38
3.5 3D hypervolume: base example 46
3.6 Slice division of a volume 47
3.7 Volume division with SDC 48
3.8 Volume division with MDC 48
3.9 Example of the bounding technique 49
3.10 Objective reordering 50
3.11 Example of local upper bounds 51
3.12 Example of slices of a volume 53
3.13 Update of the base area of a slice 54
3.14 Box partitioning of a contribution in $d = 3$ 61
3.15 Example of the box update in the computation of a contribution . . . 62
3.16 Dominated region division. 68
3.17 Data sets: linear, concave and convex 76
3.18 Data sets: wave-1, wave-3, cliff, degenerate and hard 77

4.1 Example in $d = 3$ 80
4.2 Computing the contribution of p^* to T^*. 82
4.3 Runtime performance for hypervolume indicator problems 87
4.4 Runtime performance for hypervolume indicator problem in $d = 4$ . . 87
4.5 Example of the update of contributions 90
4.6 Update of contributions when adding a dominated point 93
4.7 Runtime performance on hypervolume-contribution problems 97
4.8 Runtime performance on all contributions problems in $d = 4$ 97
4.9 Example of gHSSD algorithm in 2D 100
4.10 Example of gHSS algorithm in 2D 103
4.11 Example of the computation of the contribution $H(p, \mathrm{S})$ 105
4.12 3D example for gHSS3D algorithm 106

4.13 The three objective orders considered in gHSS3D 109
4.14 Individual contributions that have to be updated by gHSS3D 111
4.15 Computation of the joint contributions of every point in X^2 with p . . 112
4.16 Computation of the joint contributions of points in X^1 with p 114
4.17 Computation of the base area of points in X' 114
4.18 Update of the areas of points in X^1 with points in S^2 115
4.19 Example of a bad case for gHSS. 116
4.20 Approximation results for different fronts in 2D where $n = 100$ 119
4.21 Approximation results for different fronts in 3D 120
4.22 Approximation results for a bad data set for gHSSD 123
4.23 Approximation results for a bad data set for gHSS 123
4.24 Greedy and optimal subsets for the linear front, in 2D 125
4.25 Greedy and optimal subsets for several fronts, in 2D 126
4.26 Greedy and optimal subsets for the convex front, in 3D 127
4.27 Greedy and optimal subsets for the convex front, in 3D (volume left out) 128
4.28 Greedy and optimal subsets for several fronts, in 3D 129
4.29 Runtime of greedy algorithms for different settings of n and k 130
4.30 Runtime of greedy algorithms for different settings of k 131
4.31 Boxplots of Hypervolume ratio for concave data set in $d = 2$ 138
4.32 Boxplots of Hypervolume ratio for wave-2 data set in $d = 2$ 139
4.33 Boxplots of Hypervolume indicator for convex data set in $d = 3$ . . . 140

5.1 Examples of linear fronts and the corresponding $s^{(1)}$ and $s^{(2)}$. 153
5.2 Optimal investment for two sets of 30 points on a linear front 156
5.3 Optimal distribution and investment for $\mu = 7$ points 159
5.4 Extreme points included in the optimal μ-distribution for a linear front 160
5.5 Possible locations of the reference point relatively to the point $(\frac{b}{m}, b)$. 162
5.6 Examples of the optimal investment of HSR indicator over random sets 164
5.7 Examples of the optimal investment of HSR indicator over points evenly distributed 165
5.8 Examples of the change in investment with the change of location of a single point 168
5.9 Examples of the approximation of optimal μ-distributions 170
5.10 More examples of the approximation of optimal μ-distributions . . . 171
5.11 Example of solutions of a set-valued optimization problem. 173
5.12 Example of the raw fitness of set-based solutions 175
5.13 Example of set-based solutions 176
5.14 Example of the region measured to compute p_{ij} for EHSR 178
5.15 Examples of the EHSR indicator in the two-dimensional case 180
5.16 Examples of expected return 183

6.1 Summarized results for DTLZ1 in $d = 2$ 196
6.2 Summarized results for DTLZ2 in $d = 2$ 196
6.3 Summarized results for DTLZ2convex in $d = 2$ 197
6.4 Final populations for DTLZ1 in $d = 2$ 198
6.5 The 50% EAFs for DTLZ1 in $d = 2$ 198
6.6 Final populations for DTLZ2 in $d = 2$ 198
6.7 Final populations for DTLZ2convex in $d = 2$ 198

6.8 Final populations for DTLZ1 in $d = 3$ 199
6.9 The 50% EAFs for DTLZ1 in $d = 3$ 199
6.10 The 50% EAFs for POSEA on DTLZ1 in $d = 3$ 199
6.11 Final populations for DTLZ2 in $d = 3$ 199
6.12 The 50% EAFs for DTLZ2 in $d = 3$ 199
6.13 Final populations for DTLZ2convex in $d = 3$ 200
6.14 The 50% EAFs for DTLZ2convex in $d = 3$ 200
6.15 The 50% EAFs for gHSS-EMOA on DTLZ2convex in $d = 3$ 200
6.16 Percentage of dominated solutions for the DTLZ problems in $d = 2$. 201
6.17 Summarized results for the MKP in $d = 3$ with 500 items 202
6.18 Final populations for the MKP in $d = 3$ with 500 items 202
6.19 50% EAFs for the MKP in $d = 3$ with 500 items 203
6.20 Percentage of dominated solutions for the MKP with 500 items . . . 203

List of Tables

2.1 Consequences of strict indicator monotonicity 30

3.1 Convergence of $(\mu + \lambda)$-archiving algorithms 44
3.2 Algorithms for Hypervolume 59
3.3 Algorithms for computing hypervolume contributions related problems. 65
3.4 Exact Algorithms for the HSSP 70
3.5 Greedy Algorithms for the HSSP 74

4.1 Summary of the approximation ratios of the greedy algorithms for 2D 121
4.2 Summary of the approximation ratios of the greedy algorithms for 3D 121

List of Algorithms

Acronym	**Name**	**Ref.**	**Code**	**Page**
BCE3D	Bringmann, Cabello and Emmerich	[31]	[C6]	69
BF	Bringmann and Friedrich algorithm	[35]	[C1]	66
BF1	Bringmann and Friedrich algorithm ($k = 1$)	[35]	[B1]	59
B&B	Branch-and-Bound Algorithm	[75, 76]	[C7]	69
Chan	Chan algorithm	[47]	[A9]	57
DPHSS	Dynamic Programming Hypervolume Subset Selection	[3]	[C2]	66
EF	Emmerich and Fonseca algorithm	[61]	[B2]	60
EPTAS	Efficient Polynomial-time Approximation Scheme	[31]	[D5]	73
exQHV	exclusive (contributions) QHV	[119]	[B3]	60
FPL	Fonseca, Paquete and López-Ibáñez	[68]	[A3]	52
gHSS	Greedy Hypervolume Subset Selection (Incremental)	[23]	[D2]	71,101
gHSS2D	gHSS in 2D	[85, 86]	-	104
gHSS3D	gHSS in 3D	[85, 86]	-	106
gHSSD	Greedy Hypervolume Subset Selection Decremental	[23, 21]	[D1]	71,98
gHSSD3D	gHSSD in 3D	[83]	-	100
gHSSD4D	gHSSD in 4D	[83]	-	100
GSEMO	Global Simple Evolutionary Multiobjective Optimizer	[71]	[D4]	73
HBDA	Hypervolume Box Decomposition Algorithm	[102]	[A11]	57
HOY	Hypervolume Overmars and Yap	[14]	[A7]	56
HSO	Hypervolume by Slicing Objectives	[97, 131]	[A2]	52
HVC3D	Hypervolume Contributions in 3D	[83]	-	89
HVC4D	Hypervolume Contributions in 4D	[83]	-	89
HVDC3D	Hypervolume Divide-and-Conquer in 3D	[78]	[A5]	55
HV3D	Hypervolume in 3D	[15]	[A4]	52
HV3D$^+$	HV3D (improved version)	[83]	-	80
HV4D	Hypervolume in 4D	[84, 78]	[A6]	55
HV4D$^+$	HV4D (improved version)	[83]	-	86
HV4DX	HV4D alternative version	[50]	[A12]	58
HypSSP	Hypervolume SSP	[37]	[C3]	67
IHSO	Incremental HSO	[24, 25]	[B4]	61
IHSO*	Incremental HSO (least contributor)	[24]	[B6]	63
IHV3D	Incremental Hypervolume in 3D	-	-	104
ILP	Integer Linear Programming (formulation)	[100, 99]	[C4]	67
IWFG	Incremental WFG	[128, 51]	[B7]	63

Acronym	**Name**	**Ref.**	**Code**	**Page**
k-Link	k-Link (shortest path algorithm)	[100]	[C5]	68
LebMeasure	Lebesgue Measure algorithm	[62]	[A1]	51
LS	Local Search	[21, 23]	[D3]	72
QHV	Quick HyperVolume	[118]	[A9]	57
UHVC2D	Update Hypervolume Contributions in 2D	[89]	[B8]	63
UHV3D	Update Hypervolume in 3D	[84, 78]	[B5]	61
WFG	Walking Fish Group	[129]	[A8]	56

List of Acronyms

Acronym	**Meaning**
AA	Archiving Algorithm
BBOB	Black-Box Optimization Benchmarking
DA	Decision Analysis
DM	Decision Maker
EA	Evolutionary Algorithm
EAF	Empirical Attainment Function
EHSR	Extended HSR
EMO	Evolutionary Multiobjective Optimization
EMOA	EMO Algorithm
HSR	Hypervolume Sharpe Ratio
HSSP	Hypervolume SSP
KKT	Karush-Kuhn-Tucker (conditions)
KMP	Klee's Measure Problem
KP	Knapsack Problem
MCDA	Multiple Criteria Decision Aiding
MO	Multiobjective Optimization
MKP	Multiobjective KP
MSO	Multiobjective Set Optimization
PHSR	Preferability HSR
POSEA	Portfolio Optimization Selection Evolutionary Algorithm
PSP	Portfolio Selection Problem
QP	Quadratic Programming
SMS-EMOA	$\mathcal{S}$-Metric Selection EMOA
SSP	Subset Selection Problem
SVO	Set-Valued Optimization
w.r.t.	with respect to

Chapter 1

Introduction

1.1 Motivation

Everyday, people have to make decisions, conscientiously or not, over the different alternatives to solve a given task/problem. Such alternative solutions will typically have both advantages and disadvantages, some of which may be translated into objectives that one wishes to maximize or minimize. For example, when choosing the path to work, one might want to spend less time (objective 1) and travel the shortest distance (objective 2), or when booking a hotel room, one might want to find the closest one to the city center (objective 1), but minimize the money spent (objective 2) while maximizing the room comfort (objective 3). For problems like these, the ideal solution, which is the best in all objectives, seldomly exists. Indeed, objectives are typically conflicting, i.e., it may not be possible to improve in one objective without degrading another. For example, it is common that the closer a hotel is to the city center, the more expensive it is. Consequently, there is frequently more than one optimal solution.

When searching for the best option, one typically does not want to be overwhelmed with too many choices. At the same time, one wants to find the solution that suits her/him best or is, at least, good enough. From the booking services point-of-view, it should take into account that different clients have different (and typically unknown) preferences. Providing just a few options such that it is likely that one satisfies the client would be ideal. This is a subset selection problem where the booking service may have hundreds of options, but has to select a small subset to show to the client, who ultimately will decide on one solution. In a sense, one might look at the subset of options selected by the booking service as an investment in those options, where the relative amount of investment is reflected by the order in which the options are listed. This problem can thus be interpreted as the more general Portfolio Selection Problem (PSP).

In (multiobjective) optimization problems such as the above problem of planning a path to work, solutions are not typically known at hand. They have to be searched for before a set of solutions can be presented to the Decision Maker (DM). For that purpose, methods such as mathematical programming solvers and Evolutionary Algorithms (EAs) are used to generate solutions and to search for the optimal ones. The latter stand out for their ability to simultaneously search for multiple solutions and for not depending on specific problem characteristics (e.g., linearity, convexity, differentiability).

EAs also face the subset/portfolio selection problem, but multiple times. They go through many solutions, and typically cannot keep all of them. Having to choose (or invest in) some solutions over others is inevitable. A typical run of an EA consists of several iterations (called generations) each of which consists of two main steps: selection and generation of new solutions from old ones. Selection in indicator-based EAs consists in finding a subset of limited size that maximizes the indicator, which reflects the quality of a set of solutions in a scalar value. Typically, a fitness value is then assigned to every selected solution with implications on which solutions are used for generating new ones. Usually, the greater the fitness a solution has the more likely it is to be used for that purpose. In some way, this can be viewed as investing in solutions to be used to, hopefully, generate better solutions.

Together with the subset selection problem, the fitness assignment problem in EAs can be viewed as a portfolio selection problem. A recent approach aims at solving such a portfolio selection problem through the maximization of a risk-adjusted performance index, the Sharpe ratio. The initial results make this a promising approach and open up new opportunities for the integration of preferences.

Ultimately, the way an evolutionary algorithm makes choices regarding the selection of solutions, can be viewed as a reflection of some kind of preferences. Understanding these preferences, i.e., the characteristics of what is considered the best subset is very important to help ensure that the DM will find a satisfactory (set of) solution(s). For that purpose, it is essential to study in depth the methods used for solving the subset/portfolio selection problem at the core of EAs.

1.2 Summary

The central subject of this thesis is the study of the subset selection problem through the more general perspective of portfolio selection, with focus on theoretically supported methods. The first step was to work with a state-of-the-art quality indicator, and the second step was to go beyond subset selection and study a new type of indicators based on portfolio selection. In particular, this thesis aims at 1) understanding and providing an extensive overview on the theoretically supported hypervolume indicator, 2) improving the practical tools for computing problems related to the hypervolume indicator, in particular, hypervolume-based subset selection problems, and, 3) validating the use of portfolio selection in the context of EMO through Sharpe ratio maximization and providing the base tools for its theoretical and experimental analysis. The thesis document is organized as follows:

Chapter 2 – Multiobjective Optimization This chapter provides the background on multiobjective optimization and decision making and the motivation for using evolutionary algorithms in this context, with a focus on selection. Particular emphasis is given to the theoretical aspects of indicator-based selection and to the reinterpretation of selection in EAs as a portfolio selection problem.

Chapter 3 – The Hypervolume Indicator The goal of this chapter is to motivate the development of new and faster algorithms to compute hypervolume-related problems. This is achieved, firstly, by highlighting the properties of the hypervolume indicator and its importance to EMO and, secondly, by showing the difficulties associated with solving the several hypervolume-related problems arising in connection with subset selection based on the hypervolume indicator.

Chapter 4 – Hypervolume Subset Selection The central subject of this chapter is the Hypervolume Subset Selection Problem (HSSP), which consists in selecting k solutions out of a set of n solutions such that the hypervolume indicator is maximized. This chapter improves upon the state-of-the-art by proposing several algorithms for problems with a low number of objectives. In particular, new algorithms dedicated to the computation and update of hypervolume contributions lead to improvements to the computation of the HSSP in the special case where $k = n - 1$. Moreover, new algorithms for the greedy approximation of the general HSSP are proposed, and the theoretical guarantees of such approaches are studied.

Chapter 5 – Portfolio Selection This chapter considerably extends the work initiated in the paper where selection in EMO is reinterpreted as a Portfolio Selection Problem [133], and a fitness-assignment method based on the maximization of the Sharpe ratio under a given interpretation of random individual return is proposed. The positive results observed in [133] have given strength to this new perspective on selection, indicating that it is worth studying in more depth, and opening multiple paths/opportunities for further research. The main goal of this chapter is to provide theoretical support to the view of selection in EAs as a portfolio selection problem. With the theoretical analysis of the indicator proposed by Yevseyeva *et al.* [133], and the discussion of extensions of such an indicator to different types of problems, this chapter provides tools for the proposal and analysis of other indicators formalizing this view through the maximization of the Sharpe ratio.

Chapter 6 – Portfolio Selection in EMO The goal of this chapter is, firstly, to discuss the suitability of using greedy approaches to approximate the HSSP in EA selection instead of much more expensive exact algorithms, and, secondly, to discuss the suitability of indicators based on the Sharpe ratio for performing selection and fitness assignment by interpreting them together as a portfolio selection problem. An experimental study is presented, where each of these selection algorithms was integrated into a fairly simple evolutionary algorithm, and the resulting EMO algorithms were applied to a number of benchmark problems.

1.3 Contributions

The main contributions of this thesis are the following.

- An extensive review of computational problems related to the hypervolume indicator and of algorithms available to solve those problems. This review includes a rundown of the best/most adequate algorithms for each specific problem depending on the number of dimensions considered. The theoretical properties of the hypervolume indicator and of hypervolume-based selection algorithms are reviewed as well.

- New algorithms to efficiently compute and/or update the hypervolume indicator and hypervolume contributions in three and four dimensions [83]. These algorithms are shown to either match or outperform the state-of-the-art in terms of performance.

- New algorithms to approximate the HSSP in three and four dimensions using a greedy decremental approach [83]. Additionally, an approximation guarantee is derived for such an approach.

- A new algorithm to approximate the HSSP in three dimensions using a greedy incremental approach [85, 86]. An approximation guarantee is provided by connecting to known results in the literature.

- Formalization of the class of Sharpe ratio indicators and, presentation of theoretical and experimental results on a particular instance called the Hypervolume Sharpe ratio indicator [82, 81] (HSR indicator). The theoretical studies on HSR indicator cover proofs of monotonicity, independence of one of two reference points, and scaling independence under linear transformations. Moreover, it is shown that the HSR indicator always selects, at least, two points in the Pareto front (if that many exist).

- The optimal μ-distributions of the HSR indicator on two-dimensional linear fronts, and the corresponding optimal investments, are derived [81]. Such optimal μ-distributions are shown to be exactly the same as for the hypervolume indicator [81]. For other fronts, optimal μ-distribution approximations were numerically obtained and compared to the corresponding approximations for the hypervolume indicator.

- Experimental studies on the integration of the HSR indicator for both environmental selection and fitness assignment. The results in synthetic and benchmark problems show the suitability of integrating such an indicator in an EMOA.

- Description of other instances of the Sharpe ratio indicator class, one aimed at constrained optimization problems and another at problems where each solution maps to a set of nondominated points in objective space instead of a single point.

- Source code for the developed algorithms for hypervolume-based problems is made available online at `https://github.com/apguerreiro/gHSS` and at `https://github.com/apguerreiro/HVC`.

Chapter 2

Multiobjective Optimization

2.1 Optimization

Solving an optimization problem consists in finding solutions that minimize or maximize one or more objectives. Without loss of generality, minimization is assumed in this work. Formally, an optimization problem with $d \geq 1$ objectives to be minimized [109, 112] can be formulated as follows:

$$\min_{x \in \Omega} \quad f(x) = [f_1(x), f_2(x), \ldots, f_d(x)]^T \tag{2.1}$$

where Ω denotes the set of feasible points in decision space, or *feasible set*, and $f : \Omega \rightarrow \mathbb{R}^d$. The target set $\mathbb{R}^d$ is called the *objective space*. Throughout the text, subscripts are used to refer to coordinates of points or vectors (e.g., $f_i(x)$ denotes the i^{th} coordinate of vector function $f(x)$ above). The feasible set is defined as:

$$\Omega = \{x \in \mathrm{S} \mid c_i(x) = 0 \ \ \textbf{and} \ \ c_j(x) \geq 0, \ i \in \mathcal{E}, \ j \in \mathcal{I}\} \tag{2.2}$$

where S is called the *decision space*, $c_i : \mathrm{S} \rightarrow \mathbb{R}$ for all $i \in \mathcal{E} \cup \mathcal{I}$, and sets $\mathcal{E}$ and $\mathcal{I}$ are disjoint sets of integer indices referring to the *equality constraints* and *inequality constraints*, respectively. An inequality constraint c_j $(j \in \mathcal{I})$ is said to be *active* at a feasible solution x if $c_j(x) = 0$, and is said to be *inactive* if $c_j(x) > 0$. The union of the indices of all equality constraints with those of all active inequality constraints at a feasible solution x is called the *active set*, $\mathcal{A}(x)$:

$$\mathcal{A}(x) = \mathcal{E} \cup \{j \in \mathcal{I} \mid c_j(x) = 0\} \tag{2.3}$$

Single-objective ($d = 1$) and multiobjective ($d \geq 2$) optimization problems are considerably different. When the optimization problem has only one objective function, all solutions can be compared to one another with respect to their single objective value and, therefore, the feasible set is a totally pre-ordered set. On the other hand, if the problem is a multiobjective optimization problem, then the feasible set may be a partially pre-ordered set. Recall the path-to-work example from Chapter 1. If only the distance is to be minimized, solutions are easily compared/sorted based on their single objective value. If the time spent is also to be minimized, it might be the case that two solutions cannot be sorted because one takes a shorter amount of time and the other path is shorter.

2.1.1 Single-Objective Case

Optimization algorithms may or may not be able to guarantee that the solutions they return are optimal. In the single-objective case, optimality may be checked if the problem being solved belongs to a class for which appropriate optimality conditions are known, such as problems with a smooth objective function.

If a (single) objective function f and the functions c_i, $i \in \mathcal{E} \cup \mathcal{I}$, are twice continuously differentiable, then the following definitions and optimality conditions [112, p. 320-321] are applicable.

Definition 2.1 (LICQ). Given a point x^* and the active set $\mathcal{A}(x^*)$, a linear independence constraint qualification (LICQ) is said to hold if the set of active constraint gradients $\{\nabla c_i(x^*), i \in \mathcal{A}(x^*)\}$ is linearly independent.

In order to formulate the optimality conditions, the Lagrangian function is defined:

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x) \tag{2.4}$$

The following first-order necessary conditions are know as the *Karush-Kuhn-Tucker conditions* (*KKT conditions*):

Theorem 2.1 (First-Order Necessary Conditions). *Suppose that x^* is a local solution of (2.1) and that the LICQ holds at x^*. Then, there is a Lagrange multiplier vector λ^* with components λ_i^*, $i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied at (x^*, λ^*):*

$$\begin{aligned}
\nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, && && \text{(2.5a)}\\
c_i(x^*) &= 0, && \text{for all } i \in \mathcal{E} && \text{(2.5b)}\\
c_i(x^*) &\geq 0, && \text{for all } i \in \mathcal{I} && \text{(2.5c)}\\
\lambda^* &\geq 0, && \text{for all } i \in \mathcal{I} && \text{(2.5d)}\\
\lambda_i^* c_i(x^*) &= 0, && \text{for all } i \in \mathcal{E} \cup \mathcal{I} && \text{(2.5e)}
\end{aligned}$$

Since the last KKT condition (2.5e) implies that the Lagrange multipliers λ_i^* corresponding to inactive inequality constraints c_i at a local solution x^* are zero, the first KKT condition (2.5a) may also be written as:

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* \nabla c_i(x^*) = 0 \tag{2.6}$$

The first-order conditions are necessary, but not sufficient to establish that a given solution $x \in \Omega$ is optimal. That is, an optimal solution must satisfy the KKT conditions but, in general, satisfying them does not ensure optimality. Only in some particular cases are the KKT conditions also *sufficient*. For example, when the problem is convex, i.e., when the objective function and the inequality constraints are convex and the equality constraints are affine [20, p. 244].

2.1.2 Multiobjective Case

In the general d-objective case, $d \geq 2$, there may be no solution that minimizes all objectives simultaneously. Instead, there are multiple optimal solutions, each of which cannot be improved with respect to any objective without suffering degradation in another objective. The feasible set is then a partially pre-ordered set with respect to the Pareto-dominance relation. Consider two feasible solutions, $x, y \in \Omega$ and their corresponding images in objective space, $u = f(x)$ and $v = f(y)$, both in $\mathbb{R}^d$. Dominance between solutions x and y and between (the corresponding) objective vectors, or points, u and v are defined as follows [57, 92]:

Definition 2.2 (Weak dominance). A solution $x \in \Omega$ is said to *weakly dominate* a solution $y \in \Omega$, iff $u_i \leq v_i$ for all $1 \leq i \leq d$. This is represented as $x \preceq y$. Similarly, u is said to weakly dominate v, and this is represented as $u \leq v$.

Definition 2.3 ((Strict) Dominance). A solution $x \in \Omega$ is said to *(strictly) dominate* a solution $y \in \Omega$, iff $u \leq v$ and $v \nleq u$. This is represented as $x \prec y$. Similarly, u is said to dominate v, and this is represented as $u < v$.

Definition 2.4 (Strong dominance). A solution $x \in \Omega$ is said to *strongly dominate* a solution $y \in \Omega$, iff $u_i < v_i$ for all $1 \leq i \leq d$. This is represented as $x \prec\prec y$. Similarly, u is said to strongly dominate v, and this is represented as $u \ll v$.

Definition 2.5 (Incomparability). Two solutions $x, y \in \Omega$ are said to be *incomparable* iff neither $u \leq v$ nor $v \leq u$ are true. This is represented as $x \parallel y$. Similarly, u and v are said to be incomparable, and this is represented as $u \parallel v$.

Definition 2.6 (Indifference). Two solutions $x, y \in \Omega$ are said to be *indifferent*, iff both $f(x) \leq f(y)$ and $f(y) \leq f(x)$ are true. This is represented as $x \sim y$. In this case, u and v are identical ($u = v$).

Definition 2.7 (Pareto-optimality). A feasible solution $x \in \Omega$ is called *Pareto-optimal*, or *efficient*, iff there is no $y \in \Omega$ such that $f(y) < f(x)$ is true. The set of all Pareto-optimal solutions is called the *Pareto-optimal set*, while the corresponding image in objective space is called the *Pareto-optimal front*.

Definition 2.8 (Non-dominated point). If a feasible solution $x \in \Omega$ is Pareto-optimal, then $u = f(x)$ is called a *nondominated point*.

The following definitions extend the concept of dominance to sets (in objective space only) as proposed in [148], but the terminology and notation adopted have been slightly revised for greater clarity:

Definition 2.9 (Weak Set Dominance). A set $\mathrm{A} \subset \mathbb{R}^d$ is said to *weakly dominate* a set $\mathrm{B} \subset \mathbb{R}^d$ iff $\forall_{b \in \mathrm{B}}$, $\exists_{a \in \mathrm{A}} \mid a \leq b$. This is represented as $\mathrm{A} \preceq \mathrm{B}$.

Definition 2.10 ((Strict) Set Dominance). A set $\mathrm{A} \subset \mathbb{R}^d$ is said to *(strictly) dominate* a set $\mathrm{B} \subset \mathbb{R}^d$, iff $\mathrm{A} \preceq \mathrm{B}$ and $\mathrm{B} \not\preceq \mathrm{A}$. This is represented as $\mathrm{A} \prec \mathrm{B}$.

Definition 2.11 ((Strict) Elementwise Set Dominance). A set $\mathrm{A} \subset \mathbb{R}^d$ is said to *(strictly) dominate a set* $\mathrm{B} \subset \mathbb{R}^d$ *elementwise*, iff $\mathrm{A} \neq \emptyset$ and $\forall_{b \in \mathrm{B}}$, $\exists_{a \in \mathrm{A}} \mid a < b$. This is represented as $\mathrm{A} \prec\!\cdot\, \mathrm{B}$.

Definition 2.12 (Strong Set Dominance). A set $\mathrm{A} \subset \mathbb{R}^d$ is said to *strongly dominate* a set $\mathrm{B} \subset \mathbb{R}^d$, iff $\mathrm{A} \neq \emptyset$ and $\forall_{b \in \mathrm{B}}$, $\exists_{a \in \mathrm{A}} \mid a \ll b$. This is represented as $\mathrm{A} \prec\!\!\prec \mathrm{B}$.

Definition 2.13 (Set Indifference). Two point sets $\mathrm{A}, \mathrm{B} \in \Omega$ are said to be *indifferent*, iff both $\mathrm{A} \preceq \mathrm{B}$ and $\mathrm{B} \preceq \mathrm{A}$ are true. This is represented as $\mathrm{A} \sim \mathrm{B}$.

Definition 2.14 (Non-dominated point set). A set of points $\mathrm{A} = \{u_1, \ldots, u_n \in \mathbb{R}^d\}$ is said to be a nondominated point set, iff $\forall_{u,v \in \mathrm{A}}, u \neq v \Rightarrow u \parallel v$.

The widely adopted dominance relation reflects the weakest assumptions on solution comparison. Other relations could be used instead of dominance, such as the lexicographic order (prioritizes objectives) and the max-order [58]. However, using other relations encompasses stronger assumptions regarding DM preferences.

2.2 Decision Making

In multiobjective optimization, the existence of a Decision Maker (DM) is often assumed, and the optimization method is regarded as a tool to help the DM select the most preferable solution from the Pareto-optimal set. The Decision Maker is seen as an entity that has additional subjective information about the problem, known as preference information, which may allow one to discriminate among incomparable solutions. Multiobjective optimization methods may be classified into one of four classes depending on how DM preferences are used: *no preference methods*, *a priori methods*, *interactive methods* and *a posteriori methods* [110].

When no DM preferences are available at all, *no-preference methods* are used, and assumptions are made in order to arrive at a reasonable compromise solution. *A priori methods* use preference information to reformulate the problem, usually as a single-objective optimization problem. *Interactive methods* use DM preference information throughout their execution, allowing the DM to specify, and adjust, his/her preferences at each iteration, until a satisfactory solution is found. Finally, *a posteriori methods* search for Pareto-optimal solutions without using any preference information, and return to the DM the set of the best solutions found, whether they are known to be optimal or not. The DM subsequently chooses one solution according to his/her preferences.

2.2.1 Decision Analysis

Decision Analysis (DA) is the area of Operational Research that aims at helping decision makers make better choices through mathematical models. It starts with the formulation of the problem at hand. The problem is analyzed so as to identify and, formalize the criteria and constraints that should be taken into account and which characterize the possible actions (alternatives, solutions, ...) to choose from and their consequences. DA also defines the whole interaction process with the Decision Maker(s) and the formalization of their preferences. Typically, the whole process aims at providing a recommendation at the end. The problematic of providing a recommendation is typically classified in one of three types: the choice, sort and rank problematics [115]. In particular, the choice problematic is concerned with aiding with the selection of a few good actions, to help determine which actions to select and which to discard. The problematic of sorting is concerned with the classification of actions into predefined, and possibly ordered, categories. Finally, the ranking problematic is concerned with sorting all actions to provide a partial, or even, a total pre-order among them. In this case, actions are classified into classes and such classes are ranked, where actions in the same class are considered indifferent to one another.

The problem's nature may differ in several aspects, such as in criteria, Decision Makers, preferences and, uncertainty regarding any of the previous aspects [115]. In particular, problems may vary in the number of criteria (singlecriterion or multicriteria), where each criterion is interpreted as a different point of view over the problem and may be of different types (e.g. representable in ordinal or cardinal scale). The number of Decision Makers may vary. Considering just one or multiple ones has considerably different implications, for example, transitivity of preferences may not be guaranteed with multiple DMs. Preferences may be expressed in different ways. For example, as pairwise comparisons between actions, expressing preference for an action over another, or as a ranking over the different criteria. Uncertainty may lie at the level of the outcome/consequence of the available actions (risky choices [56]). For example, the action of throwing a coin in the air holds uncertainty on whether it will turn heads or tails. There can also be uncertainty regarding what the DM preferences are and whether the model chosen correctly represents those preferences.

Decision Analysis has several subfields depending on the problem characteristics. One of them is *Multiple Criteria Decision Aiding (MCDA)*, which typically covers the problems characterized by having multiple criteria, one Decision Maker, and certainty regarding the outcome of actions. MCDA pays particular attention to the interaction with the DM so as to better understand and model his/her preferences. The MCDA methods for integrating preferences can be roughly divided into two categories, those based on a *synthesizing criterion* (e.g., criteria-aggregation methods) and those based on *synthesizing preference relational system* (e.g., outranking methods). The first translates to transforming the multicriteria problem into a singlecriterion one by mapping actions into a single value, leading to a total pre-order among actions. This is typically achieved through a function that aggregates the multiple criteria and where the weight associated to each criterion reflects its importance compared to the remaining ones. MAUT (Multi Attribute Utility Theory) methods are examples of synthesizing criterion methods, where it is assumed that there is a utility function that models the DM preferences, i.e., a function that assigns a value to every action, reflecting how happy the DM is with such an action.

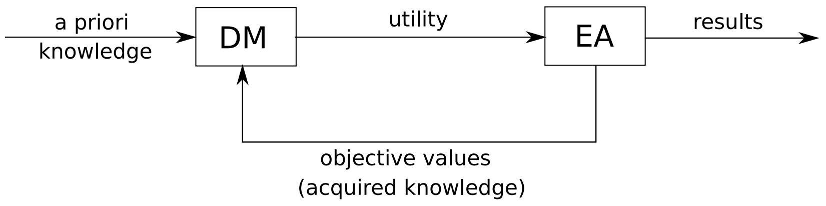


Figure 2.1: Decision Maker and Evolutionary Algorithms interaction.

The methods based on the synthesizing preference relational systems consist of successive pairwise comparisons between actions. Outranking methods are an example of such methods, which typically provide a binary relation among actions. See [115] for an overview of MCDA methods.

2.2.2 Evolutionary Multiobjective Optimization

In Evolutionary Multiobjective Optimization (EMO), Evolutionary Algorithms (EAs) are applied to multiobjective optimization (MO) problems to search for a good approximation of the Pareto front. EAs are particularly suitable in this context because of their flexibility and their ability to work simultaneously with multiple solutions. In particular, they do not depend on information such as gradients nor do they require objective functions to fulfill requisites such as being continuous or differentiable. In addition, they can provide, at once, multiple mutually nondominated solutions to the DM (see Sections 2.3 and 2.4 for more details on how EAs work).

The areas of EMO and MCDA share a common ground and somewhat complement each other, which justifies the approximation between the two communities in the last years. On the one hand, EMO has significantly contributed with methods to search for Pareto-optimal solutions while a lot of work has been done in the MCDA community towards the distinction of conflicting actions (in the optimization context, these actions represent solutions) by working on the interpretation and formalization of DM preferences. The work by Fonseca and Fleming [64, 66] shows an example of how both areas can interact.

A Decision Maker seldom has a clear and complete understanding of his/her preferences, and these may be influenced by how much he/she is (un)aware of existing feasible solutions, whether Pareto-optimal or not. Fonseca and Fleming [64, 66] presented decision making as a process where the DM and the optimization can learn from each other. The acquired knowledge can help the DM provide better informed preference information, and the EA provides better solutions according to the refined preferences. Figure 2.1 reproduces the model as presented in [64, 66].

In the model represented in Figure 2.1, the DM may provide a priori preference knowledge to the EA or not. If not, the EA may start without making assumptions about the DM preferences, and will aim at providing a good approximation of the whole Pareto front. A good approximation is typically interpreted as a set of solutions as close as possible to, and well spread along, the Pareto front. Thus, with the information provided by the EA, the DM can learn about the general tradeoff in objective space and, consequently, make better informed judgments, and refine its preferences. The DM can then supply the refined preference information to the EA that, in turn, can use it to discard nondominated solutions uninteresting to the DM that otherwise might have been considered promising, and provide more solutions that are of interest to the DM. The loop may continue until the DM is satisfied with one or more solutions presented by the EA. During such a process, the DM is free to change his/her mind over his/her preferences as he/she learns more about the solution space. Nevertheless, it is clear that the inclusion of the DM preferences within the EAs is a desirable feature when such information is available.

2.3 Evolutionary Algorithms

The first known references to Evolutionary Algorithms (EAs) were made in the 1950s [69]. Since then, many EA variants have been proposed, but they are all inspired in the idea that, in a (biological) population under natural selection, only the fittest individuals survive and reproduce, leading to increasingly fit populations with each generation. From a computational point of view, individuals can be seen, for example, as encoding possible solutions subject to some notion of quality. Individuals with higher quality are more likely to pass their genes on to the next generation. Individuals can be combined with each other and/or be perturbed slightly to generate new individuals, which is analogous to the reproduction and mutation processes that occur in nature.

Evolutionary Algorithms have been applied in several areas, such as optimization problem solving, simulation of natural and artificial systems, adaptation, and so on [73]. The general framework of Evolutionary Algorithms and their operators will be described next, in more or less general terms, while its details and implications in a multiobjective context will be discussed at the end of this section and in the subsequent one.

2.3.1 Algorithm

The structure of Evolutionary Algorithms, depicted in Figure 2.2, is characterized by the following steps: population initialization, fitness assignment, parental selection, recombination, mutation, environmental selection and checking of the termination criterion. The existing EA methodologies differ in the representation used and in the chosen operators [123]. There are three main families of methodologies: Evolutionary Programming [63], Evolution Strategies [114, 121] and Genetic Algorithms [88], which have progressively become less distinct from each other as the area continues to mature.

The various steps of a generic Evolutionary Algorithm will be explained next.

Population and Representation

The generation of the initial population is the first step of an EA. The population has a predetermined size, μ, which usually remains constant during EA execution, and the initial individuals are usually generated at random. Each individual may be characterized by a genotype and a phenotype. The genotype encodes the characteristics of the individual, for example, as a binary string. The phenotype is the corresponding object in the space of the original problem. Note that a given genotype decodes to a single phenotype, but the opposite may not be true. For example, the integer solution 4 may be encoded as 0100, so 0100 is the genotype and 4 the phenotype. A representation also includes the mapping from the genotypes to the

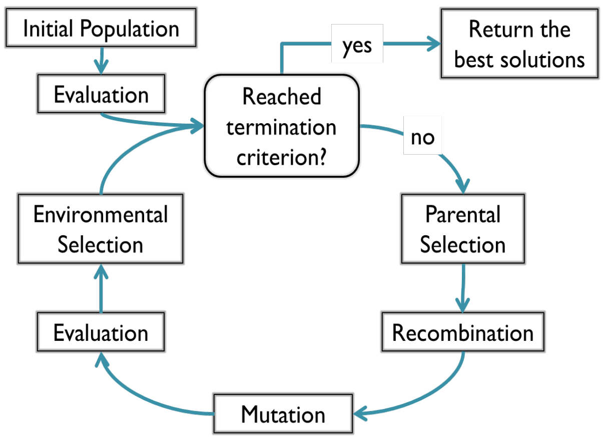


Figure 2.2: Evolutionary Algorithm.

phenotypes. Genotypes may be represented as binary strings, real-valued vectors, sequences of instructions, discrete structures such as permutations and graphs, or combinations of any of these, and be of fixed or variable length [59].

The representation should be chosen based on the problem and on the recombination and mutation operators. These operators are the ones responsible for defining the individual's neighborhood, which influences the EA's performance, while the representation has impact at the implementation level and on execution time.

Fitness Assignment

After generating the initial population, individuals are evaluated according to a quality measure, and are subsequently assigned a real fitness value through fitness assignment [59]. The function that maps individuals to a fitness value is called the *fitness function*. The higher the quality of an individual according to some problem-dependent notion of quality, the higher the fitness should be. Frequently, the assigned fitness also reflects other aspects such as those related to how much an individual differs from others in the population and/or how much (quality) it adds to the population. Also, some of the individuals may not be valid individuals. For example, in a constrained optimization problem, an invalid individual would be one that does not satisfy all problem constraints. If invalid individuals may be generated by the algorithm, it is common practice to penalize them at this stage by assigning them a low fitness value.

Parental Selection

The following phase is parental selection and is problem-independent. This operator selects individuals according to their fitness to obtain the parents that will be recombined to generate new individuals. Selection may be performed through probabilistic or deterministic mechanisms, but the fittest individuals should have higher chances of being selected for breeding. There are several sampling methods, such as roulette-wheel selection [73], tournament selection [74] and Stochastic Universal Sampling (SUS) [11], and so on. One important characteristic of selection (understood as the combination of fitness assignment and sampling mechanisms) is selection pressure. The stronger the selection pressure is, the more the fittest

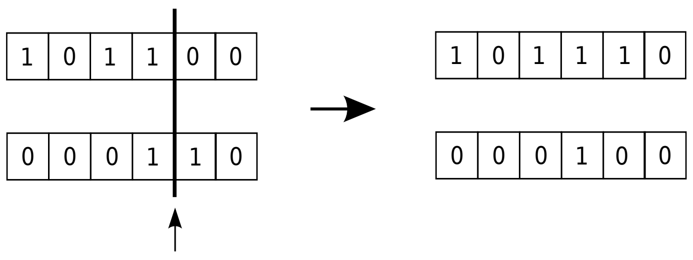


Figure 2.3: A crossover example.

individuals will be selected over weaker individuals, which may lead to premature convergence. On the other hand, if selection pressure is too low, more promising solutions will be exploited at very low rates, and the population may not converge at all.

Recombination and Mutation

Recombination and mutation are variation operators which lead to the generation of new individuals. Depending on the EA instance, either one or both operators may be used. When both operators are used, there is usually a high probability of applying recombination, whereas mutation is typically applied with low probability. Typically, in the case of recombination, two individuals, (the parents) are randomly picked using some selection method, and are recombined to generate new solutions. Examples of the recombination (crossover) operator are the single-point and multi-point crossover, among many others. The choice of this operator depends on the problem, and its implementation depends on the representation used. An example of the single-point crossover operator is depicted in Figure 2.3, where solutions are encoded as binary strings. It consists in choosing a cutting point in each individual (if they are fixed-length then the cut point is the same), and generating two new individuals, concatenating the first part of the first parent with the last part of the second, and vice-versa.

The mutation operator is an operator that usually introduces small changes in a candidate solution, but it may also be a very important operator. There are different points of view on the mutation operator [59]. Some authors look at mutation as a complementary operator to recombination, whose goal is to perturb the solution slightly in order to introduce small amounts of diversity. The other point of view looks at mutation as a powerful mechanism by itself, avoiding the need for a recombination operator.

Environmental Selection

After recombination and/or mutation, the new individuals are evaluated and fitness is assigned to them. After this phase, the environmental selection occurs. This operator determines which individuals among those currently in the population move to the next generation together with which offspring. Two common environmental selection methods are represented as (μ, λ) and $(\mu + \lambda)$, where μ is the population size and λ is the number of new individuals generated at each generation. This notation was introduced in Evolution Strategies [18]. The first selection method,

(μ, λ), assumes that $\lambda > \mu$ and replaces the current generation by μ of the new λ individuals. The second selection method, $(\mu + \lambda)$, passes on to the next generation μ individuals chosen among the individuals in the population and the new ones. The way in which the μ individuals that will continue to the next generation are selected may vary. The best ones may be chosen, or a tournament selection or other technique may be used. $(\mu + \lambda)$ selection has the important characteristic of enabling the algorithm to always preserve the best solutions found throughout the generations. It is therefore, an elitist strategy.

Termination criteria

The algorithm runs until a termination criterion is met. This criterion may be the number of past iterations, or the accumulated number of generated offspring (for optimization problems this is sometimes called the number of function evaluations), or some level of quality that individuals must reach, the number of iterations without improvement, or even some measure of (the lack of) population diversity.

On the choice of operators

If the representation and, in particular, the operators are not chosen carefully, the algorithm may converge prematurely. The population may become very similar and stagnate, preventing other important regions of the search space from being explored. Moreover, the representation and the operators used may strongly influence the population's ability to evolve. For example, in an optimization context, the algorithm may never be able to reach the optimal solution if the operators do not guarantee that the whole decision space can be reached.

Although new EAs are frequently proposed as a specific combination of operators, there is freedom to replace operators. Such modifications may improve an EA performance for some problems. Due to the amount of combinations of operators and of parameter settings (e.g., combination/mutation rate) it is hardly possible to ensure the best combination for each problem instance. With the availability of tools for the automatic design of EAs [19] and the growing theoretical knowledge on parameter setting [55], more focus can be given to the design of operators and on the study of their properties in order to understand their advantages and their limitations. The subsequent work of finding an appropriate combination of operators and parameter settings for each problem may be, carefully, left for such automatic design tools to decide.

2.3.2 Evolutionary Multiobjective Optimization Algorithms

In Evolutionary Multiobjective Optimization Algorithms (EMOAs), each individual typically represents a solution and thus, by working with a set of solutions (population), EAs have the ability to simultaneously explore different regions of the objective space. This is particularly desirable in multiobjective optimization because it allows to find multiple mutually nondominated solutions that approximate, in a single run, the Pareto front. Although only a finite subset of the Pareto front may be found by the EMOA, that is typically the end goal. When the Pareto front is too large or even infinite, the task of finding the whole Pareto front becomes unbearable and it is usually undesirable for the DM to have to go through too many solutions. Hence, the goal of EMOAs is to find a representative and finite subset of Pareto-optimal solutions. To do so, maintaining a good and diverse set of solutions is crucial. However, due to the conflicting nature of objectives, that task is not trivial, as having to choose between mutually nondominated solutions is seldom avoidable.

Preference Information

The quality of the approximations produced by an EMOA is highly affected by the, either explicit or implicit, preferences reflected in the fitness assignment and the environmental selection, as these operators are determinant to the EAs ability to explore regions of interest and to retain the best solutions. Fitness assignment is part of the evaluation of solutions. Such evaluation can be viewed as a three-phase process: evaluate each solution into objective values, evaluate solution cost/value based on such objective values to reflect the utility of the solution to the DM and, assign fitness based on such a cost/value [66]. The three evaluations reflect how good a solution is from three different perspectives: from the problem, the DM and the search process points of view. Sometimes the objective value (in single-objective optimization) or the cost is directly used as fitness. Environmental selection defines which solutions are more worth keeping, which is typically related to their quality.

Whenever preference information is available, either provided by a MCDA technique and/or based on domain knowledge, it can be integrated into the EA to help decide which nondominated solutions to favour. When no (explicit) preference information is available, only weak assumptions can be made with certainty, such as the fact that a dominated solution should never be preferred over a nondominated one, or that any feasible solution is preferable to an infeasible one. However, even these assumptions can be viewed as preference information. Different (explicit/implicit) preference information and different methods to integrate preferences leads to (possibly) different strategies and (possibly) different outcome solutions. Some work has been done towards the integration of preference information in EAs [146, 147, 66]. The *preferability* relation [66] is an example of a decision making framework that can be integrated in EAs, which allows to accommodate weak assumptions and other types of relational preference information through objective/constraint prioritization and/or the establishment of goals. See [28] for an up-to-date overview of how preference information has been included in EMOAs.

Consider the example of using the preferability relation to reflect preferences over solutions of constrained optimization problems [66]. By specifying a high priority vector (on constraints) and a low priority vector (on objectives) as suggested in [66], pairs of solutions are compared first according to constraint values and only if both satisfy all constraints are they compared based on objectives. An (infeasible) solution is preferable to another solution if 1) the former is better than or equal to the latter in all of its unsatisfied constraints, and is strictly better in at least one of them, or 2) both solutions are equal in the constraints which the former violates, and the latter violates more constraints or is dominated by the former with respect to the objectives. Thus, a feasible solution is always preferable to an infeasible one. The comparison of feasible solutions is equivalent to Pareto dominance relation. Incomparable feasible solutions remain incomparable under such a preferability relation and so do infeasible solutions failing to satisfy the above criteria of comparability. Hence, a pre-order on feasible and infeasible solutions is induced.

An example of the incorporation of the preference information reflected by the preferability relation in an EA is to assign a cost to each solution reflecting how many solutions are preferable to it, thus providing a total order among solutions. Consequently, in the above example, better (i.e., lower) cost is given to feasible solutions over infeasible ones. Such costs are taken into account in the fitness assignment, whereas environmental selection may use it, for example, to rank solutions and subsequently select the better ranked ones. Thus, fitness assignment may provide higher chances of reproduction to feasible solutions and environmental selection can ensure that infeasible solutions are selected only after all feasible ones are selected. Hence, the EMOA reflects a preference for generating solutions in the neighborhood of, and keeping, the nondominated (feasible) solutions. However, the equal cost (zero) assigned to nondominated feasible solutions reflects indifference between them and thus, selection in such a case is somewhat random, which may result in a poorly spread approximation.

With or without DM preference information, it is frequently hard to have enough information to resolve all incomparabilities and thus, have a total order on solutions that is certainly in agreement to the DM. To attenuate the lack of complete knowledge of preference information, EMOAs typically aim at maximizing their chances of satisfying the DM by seeking a good and diverse (in objective space) set of solutions. Towards this goal, EMOAs have to resolve the (remaining) incomparabilities, particularly in selection by deciding which nondominated solutions to keep/discard, and how this is done is crucial for EMOA success. Such decisions go beyond DM preference information and may be viewed as EMOA preferences. Thus, it is important to analyze selection methods to understand the EMOAs inner preferences.

2.4 Selection in EMOAs

In general, multiobjective EAs are developed with two main goals in mind: searching for the best solutions towards the Pareto front, and searching for a diverse set of solutions [142]. The notion of diversity is subjective, and depends on the DM preferences. However, when these are not known, it is generally understood as a set of solutions well (e.g., evenly) spread along the Pareto front. If only the first goal is considered, the EA may converge to a small region of the front. If only the second goal is considered, the algorithm may not be able to approach the Pareto front at all. Traditionally, these two goals were considered separately. For example, fitness assignment would usually assign fitness according to some notion of individual quality and then these fitness values would be modified in order to penalize solutions in crowded regions given some notion of neighborhood. Subsequently, indicator-based approaches were proposed, focusing on the population quality as a whole, combining individual quality and population diversity into a single measure.

Even if additional preference information is not available, the fitness assignment and environmental selection implicitly induce preferences [147]. The methods used in the literature either induce these preferences over solutions (through methods to evaluate solutions and optionally using a diversity preservation technique), or over sets (through quality indicators), or by decomposing the multiobjective problem into a finite set of single-objective ones (decomposition methods). More recently, a new method that looks at environmental selection and fitness assignment as a portfolio selection problem [133], tackles both tasks at once, which somewhat induces preferences both over sets and over solutions. The different selection methods and the theoretical properties that characterize their inner preferences are discussed next.

2.4.1 Methods Focused on Individual Quality

The main goal of EAs is to improve the individuals of the population at each generation. This is performed by giving higher chance of reproduction to the most promising solutions, expecting that those can lead to even better solutions. The main EA components responsible for achieving this goal of searching towards better individuals are fitness assignment and selection. However, in the presence of multiple conflicting objectives, it is not easy to define what a good and accurate measure of individual quality is. EAs use a fitness scalar to indicate how promising an individual is, which suggests the need for a function to map each objective vector onto a scalar.

In this subsection, the three main methods used to drive the population towards better solutions are explained. Aggregation-based and Pareto-based methods work by defining suitable fitness assignment strategies. Population-based methods typically apply different selection strategies to obtain different subsets of populations.

Aggregation-based methods

The most obvious way to assign a scalar value to an objective vector is by linearly aggregating the objective values. Therefore, the multiobjective problem is transformed into a single-objective optimization problem in the following way:

$$\min_{x\in\Omega} \sum_{i=1}^{d} w_i f_i(x) \tag{2.7}$$

where $w_i \geq 0$, and at least one of the weights is strictly positive. Usually, objective functions are normalized in some way, and assigning weights to each objective usually requires prior knowledge. In the literature, there are several alternative methods to aggregate objective values, namely [65]:

1. Target vector optimization
2. Goal attainment
3. Multiple Attribute Utility Analysis (MAUA)
4. Constraint handling with penalty function
5. Many others.

For instance, target vector optimization consists of defining a goal for each objective and assigning fitness according to the distance from solutions to this goal vector, where this distance should be minimized and different notions of distance can be used. The general disadvantage of aggregation-based methods is the need for prior knowledge in order to set appropriate weights and/or goals, which is often not an easy task.

Population-based methods

Population-based [142] (also known as Criteria-based) methods rely on selecting different individuals based on different fitness functions at each iteration. In the simplest case, each fitness function accounts for each of the objectives, so that a different objective is used to evaluate individuals at each time. The classical example of a population-based method is VEGA [120] which divides the mating pool into d equal-sized parts, and each one is filled using a different objective. Other methods are surveyed in [142, 65]

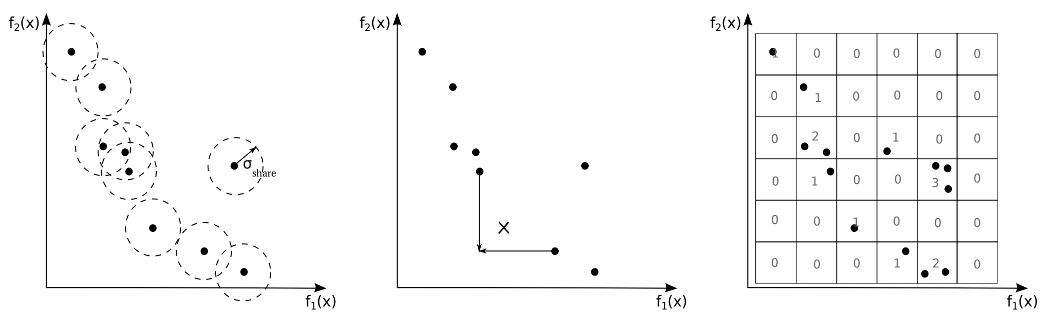
Pareto-based methods

Pareto-based approaches [65] rely on the notion of Pareto dominance to assign fitness to individuals. For example, fitness may be assigned based on the number of solutions which dominate or are dominated by each individual. For example, Fonseca and Fleming [64, 65] proposed a method based on the number of individuals that dominate each solution (*dominance rank*). In this case, since the nondominated points have no points dominating them, the nondominated points have the highest fitness values. Another method is the *nondominated sorting* used in NSGA-II [53], also known as *dominance depth* where solutions are ranked in classes in the following way. Every nondominated solution is assigned rank 0 and then removed, those solutions among the remaining ones that become nondominated are assigned rank 1 and, also removed. This procedure is repeated until there are no more solutions to rank. Lastly, there is the *dominance strength* used by SPEA2 [143] where the strength of an individual is defined to be equal to the number of solutions it dominates. In SPEA2 the (raw) fitness of an individual is computed based on the sum of the strength of the individuals that dominate it. In such a case, the lower the raw fitness the better.

One advantage of Pareto-based approaches is that, in contrast to some aggregation-based methods, they are not sensitive to non-convex Pareto fronts. Although Pareto-based methods correctly privilege nondominated solutions, they are prone to genetic drift. Genetic drift occurs when several solutions have the same fitness value, as it happens among nondominated solutions. This may cause the algorithm to concentrate only in a small region of the Pareto front due to stochastic errors in selection. Note that the described Pareto-dominance methods partition the population in dominance classes [147] and, alone, do not distinguish solutions in the same dominance class.

2.4.2 Diversity Preservation Techniques

When the methods to guide the EA search described in the previous subsection are used without any auxiliary mechanisms, they tend to converge to a limited part of the Pareto front. To address this, several diversity preservation techniques may be used. This subsection is dedicated to the description of some common diversity preservation techniques, most of which include what can be seen as population density information. This information is used to penalize individuals in more densely populated regions of the decision or objective space. The higher the density in the region where an individual is located is, the lower are its chances of being selected. Hence, most diversity preservation techniques fall into one of the following



(a) Fitness sharing (b) crowding distance (c) histogram

Figure 2.4: Diversity preservation techniques.

three statistical density estimation related categories [142]: Kernel methods, Nearest Neighbor techniques and Histograms. These techniques, will be briefly explained next.

Kernel methods

Kernel methods are based on a so-called kernel function, which is function of a distance between two individuals. Kernel methods compute the density estimate by summing the values of the kernel function given the distances of an individual to all other individuals.

The most popular diversity preservation technique of this type is fitness sharing, which defines a niche as a region of radius σ_{share} around each individual. See Figure 2.4(a) for an example. The density estimate for an individual p is computed based on the distances between individual p and all individuals inside its neighborhood only, normalized by σ_{share}. After the density estimate is computed, individual fitness is penalized, by dividing it by the density estimation. The underlying idea is that an individual fitness is shared with its neighbors. The main disadvantage of fitness sharing is the need to set σ_{share}.

Nearest neighbor

As the name says, in the nearest neighbor method, the density estimator around an individual is based on the distance to its k-nearest neighbors. The most popular technique of this type is Crowding. One example of crowding in Multiobjective EAs [53] is one that sums the crowding distance for each objective. The crowding distance for each objective is the normalized difference between the closest point with a higher value, and the closest point with a lower value of that objective. See Figure 2.4(b) for an example. Deb *et al.* [53], use this crowding distance to untie solutions with the same (dominance depth) rank in a tournament, where the winner is the one with higher crowding distance. The advantage of this method with respect, for example, to fitness sharing, is that it may be easier to set k than σ_{share}.

Histograms

Histogram methods [96, 142] consist of dividing the objective space into equal-size rectangular non-overlapping regions, called cells or boxes. The neighborhood of a point is therefore restricted to the box where it is located. The density of the box is defined by the number of individuals in that box. Figure 2.4(c) shows an example.

There are histogram variants where the box size is adjusted during the algorithm's execution, according to the current minimum and maximum value of each objective. Histogram methods have the advantage of being computationally more efficient than with Kernel and nearest neighbor methods.

2.4.3 Decomposition Methods

Decomposition methods [135] decompose the multiobjective optimization problem into multiple single-objective optimization problems. One way to do that is by defining a finite set of aggregation functions. For each of these functions, the solution in the population that minimizes it is selected to be part of the next generation. Therefore, if the parameters of the aggregating functions are appropriately set, diversity is implicit, and the algorithm tends to find good approximations that are also reasonably well distributed.

2.4.4 Methods Based on Set Quality

A popular trend in multiobjective evolutionary algorithms is the use of quality indicators to support selection [141, 16]. Quality indicators first appeared as methods for the performance assessment of EMOAs and soon, quality indicator-based selection became state-of-the-art selection methods in EMOAs. This was due to the ability of quality indicators to explicitly favor a combination of the best solutions and diversity, in a cooperative way, and due to their (theoretical) properties.

Quality indicator-based selection

A set-quality indicator is a function that maps sets of points in $\mathbb{R}^d$ onto real values [140]. In the context of indicator-based EMOAs, sets of bounded size maximizing or minimizing some quality indicator are of interest. In $(\mu + \lambda)$ environmental selection, for example, an optimal subset of μ new parents is sought among all subsets of the current population of μ parents and λ offspring. More generally, indicator-based EMOAs search for a bounded-size subset X of the feasible set Ω, such that the corresponding image in objective space $f(\mathrm{X})$ is an optimal subset of $f(\Omega)$ with respect to a set-quality indicator given a maximum subset size. Without loss of generality, indicator-based subset selection [10, 37] can be formulated as follows:

Problem 2.1 (Subset Selection Problem (SSP)). Given a set $\mathrm{S} \subset \mathbb{R}^d$, an integer $k \geq 0$, and a quality indicator $I : 2^{\mathbb{R}^d} \to \mathbb{R}$, find a discrete point subset $\mathrm{A} \subseteq \mathrm{S}$ such that $|\mathrm{A}| \leq k$ and

$$I(\mathrm{A}) = \max_{\substack{\mathrm{B} \subseteq \mathrm{S} \\ |\mathrm{B}| \leq k}} I(\mathrm{B})$$

In multiobjective optimization, the values assigned to sets by a quality indicator are typically expected to reflect the proximity of the points in those sets to the Pareto front, as well as the extent and uniformity of their distribution in objective space. Furthermore, quality indicators should not contradict Pareto dominance,

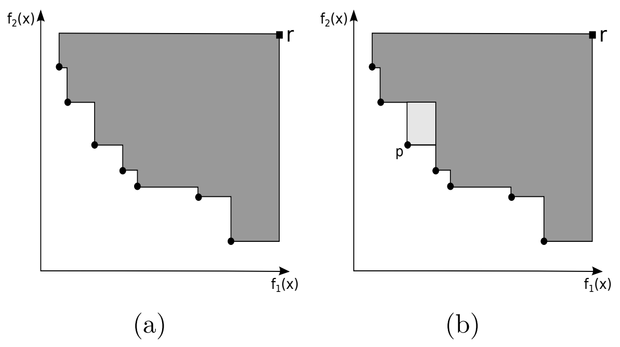


Figure 2.5: Hypervolume Indicator.

but incomparable sets can be assigned seemingly arbitrary values. Therefore, quality indicators can be understood as preference models allowing any two sets to be compared and, as long as the corresponding indicator values are different, the preferred one to be identified. If the indicator values are equal, the sets are said to be indifferent with respect to those preferences.

The hypervolume indicator [68] and the additive ϵ-indicator [141] are examples of set-quality indicators known to be monotonic with respect to Pareto dominance [140] which ensures that selection favours dominating over dominated solutions. This and other properties of set-quality indicators help understand their guarantees and limitations when used for selection in EMOAs. The properties of indicator-based selection methods are discussed in Section 2.4.6.

Hypervolume-based selection methods The Hypervolume Indicator (also known as $\mathcal{S}$-metric or Lebesgue measure) is probably the most widely used quality indicator in Evolutionary Multiobjective Optimization. It is defined as the measure of the region dominated by the points in a nondominated point set and bounded above by a reference point r. This region is the shaded region in Figure 2.5(a). The contribution of a point is the measure of the region exclusively dominated by that point. Figure 2.5(b) shows the contribution of the point p in light gray.

A popular EA based on hypervolume-based selection is SMS-EMOA [16]. SMS-EMOA uses a $(\mu + 1)$ strategy, i.e., it maintains a μ-size population, generates a new individual at each iteration, and the best μ individuals among the old μ individuals and the new individual are chosen to become part of the next generation. Given the $\mu + 1$ individuals, the selection of the μ individuals of the next generation is performed by sorting individuals in classes using nondominated sorting (see section 2.4.1 on Pareto-based methods) and then removing the individual that contributes the least hypervolume to the set of worst ranked solutions. A variant of this selection method consists of removing the least contributor only if all individuals are mutually nondominated (they are all assigned the same rank). If not, then the individual that became dominated first (more generations ago) is the one that is excluded [89]. SMS-EMOA achieves similar results with either technique.

No matter which of the two hypervolume-based selection methods explained above is used, it is certain that the algorithm only includes a newly generated point in the next generation if it is not worse than the current worst individual in the population. Note that both methods have different notions of what is the worst individual in the presence of dominated points, but when the population only contains nondominated points, both agree on the worst being the one that contributes less hypervolume. This procedure ensures that, if the reference point is kept unchanged,[1] the hypervolume indicator of the nondominated solutions in the population never decreases.

There are other selection methods based on the hypervolume indicator. For example, Zitzler and Künzli [141] proposed a general indicator-based evolutionary algorithm (IBEA) before SMS-EMOA. One of the indicators tested in IBEA was a binary version of the hypervolume indicator, i.e., an indicator based on the standard unary hypervolume indicator that assigns a real value to an ordered pair of individuals. This binary indicator is used to compute the fitness of each individual. DM preferences may be included in EMOAs using the alternative weighted hypervolume indicator [138], where preference information is provided through a weight distribution over the objective space.

Other indicator-based selection methods Most of the set-quality indicator-based selection methods are based on the hypervolume indicator, but other alternatives exist. For example, Zitzler and Künzli [141] proposed the use of a binary indicator, the additive ϵ-indicator. Given two points u and v, the additive ϵ-indicator of the pair (u, v) is the minimum value that must be added to all components of v in order for u to weakly dominate point v (assuming minimization). The authors report that the IBEA algorithm with a fitness assignment function based on the additive ϵ-indicator (and also with the binary hypervolume indicator) is able to outperform other more traditional EAs that seek converge and diversity in two distinct steps. Another example is the selection based on the contribution to the R2-indicator [43, 124], an indicator that allows preference information to be accounted through the definition of multiple utility functions.

2.4.5 Portfolio Selection Approach

Portfolio selection [108] can be seen as a generalization of subset selection that consists in assigning non-negative weights to all elements of the parent set so as to maximize some function, or functions, of those weights. This class of problems commonly arises in finance (e.g., in the stock market), where the parent set is a set of assets in which to potentially invest, each weight represents the amount of capital to be invested in the corresponding asset, and the function(s) to be optimized model(s) the financial performance of the resulting portfolio given the performance of the individual assets and the amounts invested in them.

In the classical Markowitz formulation [108], the performance of individual assets is modelled by means of random variables representing their (uncertain) financial return, and it is assumed that the corresponding expected values, variances and covariances are either known or can be estimated. Similarly, portfolio performance is assessed in terms of expected return, to be maximized, and return variance, to be

[1]In the original version of SMS-EMOA [16] the reference point is updated at each generation (only for problems in $d \geq 3$ dimensions), and thus the hypervolume indicator of the population may decrease during the run.

minimized. Formally:

$$\max_x \quad \sum_{i=1}^{n} r_i x_i = r^T x \tag{2.8a}$$

$$\min_x \quad \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j = x^T Q x \tag{2.8b}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = 1, \quad x_i \in [0,1],\ i = 1, \ldots, n \tag{2.8c}$$

where n is the number of assets, r_i denotes the expected return of asset i, and the covariance of the returns of assets i and j is denoted by q_{ij}. The unknown solution is represented by $x = (x_1, \ldots, x_n)^T$, where each x_i denotes the fraction of the available capital to be invested in asset i.

Typically, assets with high expected return entail higher risks, and low risk assets have low expected return. Investing in similar assets usually leads to a high variability and therefore higher risk, while distinct assets have low variability. It is thus advisible not to invest only in high expected return assets nor in very similar assets. The investor has to find a portfolio that balances the risk and the overall expected return and only through diversification of assets is it possible to reduce the risk for a given expected return.

Since there are two objectives in the above formulation, there is generally a set of Pareto-optimal portfolios within which higher expected return can only come at the expense of higher return variance, i.e., higher risk. Therefore, an investor must select a suitable compromise solution such that the expected return compensates for the risk taken. An also classical formalization of this idea is the performance index known as reward-to-volatility ratio, or Sharpe ratio [49], the maximization of which leads to an efficient risk-balanced portfolio:

Problem 2.2 (Sharpe-Ratio Maximization). Let $\mathrm{A} = \{a^{(1)}, \ldots, a^{(n)}\}$ be a non-empty set of assets, let vector $r \in \mathbb{R}^n$ represent the expected returns of these assets and let matrix $Q \in \mathbb{R}^{n \times n}$ be the covariance matrix of asset returns. Let each component x_i of the investment vector $x \in [0,1]^n$ denote the investment in asset $a^{(i)}$. Find a global solution, x^*, of:

$$\max_{x \in [0,1]^n} \quad h(x) = \frac{r^T x - r_f}{\sqrt{x^T Q x}} \tag{2.9a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = 1 \tag{2.9b}$$

where r_f represents the return of a baseline riskless asset, and $h(x)$ is the Sharpe Ratio of x.

Although Problem 2.2 is a non-linear programming problem and may not be easy to solve, by homogenizing $h(x)$ it can be restated as an equivalent convex quadratic programming (QP) problem provided that Q is positive definite [49]:

Problem 2.3 (Sharpe-Ratio Maximization – QP Formulation).

$$\min_{y \in \mathbb{R}^n} \quad g(y) = y^T Q\, y \tag{2.10a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} (r_i - r_f)\, y_i = 1 \tag{2.10b}$$

$$y_i \geq 0, \quad i = 1, \ldots, n \tag{2.10c}$$

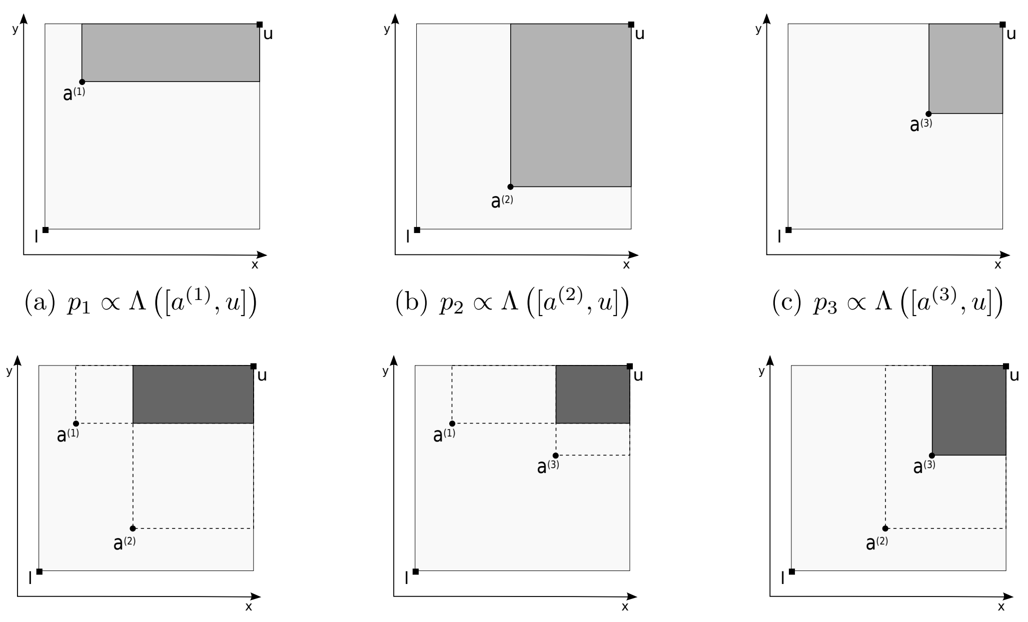
The two problems are equivalent in the sense that the optimal solution x^* of Problem 2.2 can be determined from the optimal solution y^* of Problem 2.3 ($x^* = y^* / \sum_{i=1}^{n} y_i^*$), and vice-versa.

So far, the set of assets A was considered to be fixed, and therefore both r and Q were constant. In this work, the notation $h^{\mathrm{A}}(x)$ and $g^{\mathrm{A}}(y)$ will be used instead of $h(x)$ and $g(y)$, respectively, to highlight the dependence of these functions on the set of assets when it is allowed to vary.

Yevseyeva *et al.* [133] noticed that the Portfolio Selection can be easily related to Evolutionary Algorithms in the following way: individuals are seen as assets and the expected return of an asset/individual can be related to individual quality as the probability of satisfying the DM preferences. Moreover, similar individuals have a high risk associated with them, since it is likely that if one does not satisfy the DM, neither will the other. Fitness can then be seen as the investment in an individual. Therefore, the goal of an EA is to invest in the individuals (assign fitness) in such a way that expected return is maximized (good individuals are chosen for reproduction) and risk is minimized (lack of diversity is avoided). Thus, with a suitable interpretation of return of an individual, the expected returns (vector r) and covariances between individuals (matrix Q) can be estimated. Environmental selection and fitness assignment problems can then be directly interpreted as a portfolio selection problem and solved, for example, through the Sharpe-ratio maximization problem (Problem 2.2).

Still according to Yevseyeva *et al.*'s [133], the return of each individual in an EMOA population depends on the preferences of a Decision Maker (DM). It is further assumed that such preferences are not fully known in advance and that the associated uncertainty can be described by some probabilistic model. As a result, individual (asset) returns are random variables whose expected values and variances/covariances can be computed or estimated in some way. Finally, returns are assumed to be additive across individuals.

This interpretation of an individual's return, lead to a formulation where expected return and covariances are computed based on the hypervolume indicator. DM preferences are expressed in terms of a goal vector, not known in advance, and a solution is considered acceptable if the corresponding objective vector weakly dominates that goal vector. Otherwise, it is considered unacceptable. The expected return of a solution is the probability of that solution being acceptable to the DM, assuming a uniform distribution of the DM's goal vector in an orthogonal range $[l, u] = \{x \in \mathbb{R}^d \mid l \leq x \leq u\}$, where $l, u \in \mathbb{R}^d$ are such that $l \ll u$, and d denotes the number of objectives. For the ith individual in a population $\mathrm{A} = \{a^{(1)}, \ldots, a^{(n)}\} \subset \mathbb{R}^d$, this is represented by component p_i of a vector p, whereas the return covariance between the ith and jth individuals is represented by element



(a) $p_1 \propto \Lambda\left([a^{(1)}, u]\right)$ (b) $p_2 \propto \Lambda\left([a^{(2)}, u]\right)$ (c) $p_3 \propto \Lambda\left([a^{(3)}, u]\right)$

(d) $p_{12} \propto \Lambda\left([a^{(1)}, u] \cap [a^{(2)}, u]\right)$ (e) $p_{13} \propto \Lambda\left([a^{(1)}, u] \cap [a^{(3)}, u]\right)$ (f) $p_{23} \propto \Lambda\left([a^{(2)}, u] \cap [a^{(3)}, u]\right)$

Figure 2.6: Examples of the regions measured to compute p_i and p_{ij} given a point set $\mathrm{A} = \{a^{(1)}, a^{(2)}, a^{(3)}\} \subset \mathbb{R}^2$ (from [81]).

q_{ij} of a matrix Q $(i, j = 1, \ldots, n)$. Let:

$$p_{ij}(l, u) = \frac{\Lambda([l, u] \cap [a^{(i)}, \infty[\cap [a^{(j)}, \infty[)}{\Lambda([l, u])} = \frac{\prod_{k=1}^{d} \max(u_k - \max(a_k^{(i)}, a_k^{(j)}, l_k), 0)}{\prod_{k=1}^{d}(u_k - l_k)} \tag{2.11a}$$

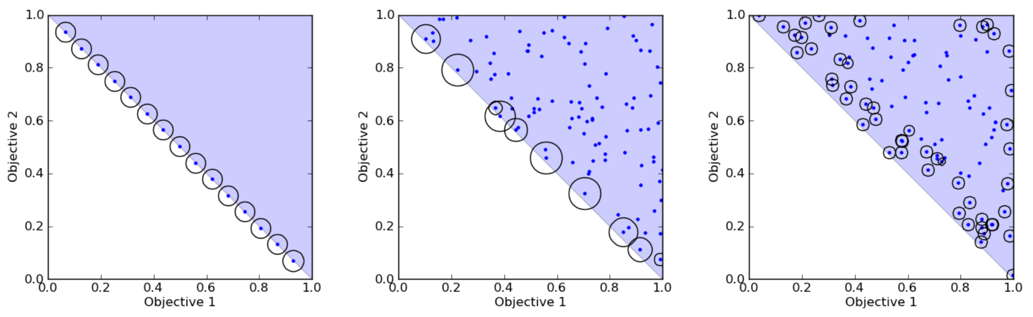
$$r_i(l, u) = p_i(l, u) = p_{ii}(l, u) \tag{2.11b}$$

$$q_{ij}(l, u) = p_{ij}(l, u) - p_i(l, u)\, p_j(l, u) \tag{2.11c}$$

where $l, u \in \mathbb{R}^d$ are two reference points and $\Lambda(\cdot)$ denotes the Lebesgue measure [16] of the region in the argument. Note that p_{ij} is, therefore, the normalized hypervolume indicator of the region jointly dominated by $a^{(i)}$ and $a^{(j)}$ inside the region of interest, $[l, u]$. For the sake of readability, $P = [p_{ij}]_{n\times n}$ and $Q = [q_{ij}]_{n\times n}$ will be assumed to have been previously calculated and, therefore, parameters l and u from expressions (2.11) will be omitted as long as no ambiguity arises. It follows from the definition of q_{ij} that $Q = P - pp^T$.

Assuming without loss of generality that $l = (0, 0)^T$ and $u = (1, 1)^T$, and thus $\Lambda([l, u]) = 1$, the areas of the shaded regions in Figures 2.6(a) to 2.6(f) are exactly p_1, p_2, p_3, p_{12}, p_{13} and p_{23}, respectively. Note that p_i is the area of the subregion of $[l, u]$ that is dominated by $a^{(i)}$, while p_{ij} is the area of the subregion of $[l, u]$ simultaneously dominated by $a^{(i)}$ and $a^{(j)}$. Since, as depicted in Figure 2.6(f), $a^{(3)}$ is dominated by $a^{(2)}$, it follows that $p_{23} = p_3$. By observing Figure 2.6 note that, the more similar two solutions are in the objective space, the higher will be p_{ij} and their covariance, whereas the more distinct they are, the lower will be p_{ij}, and so will their covariance.

On the other hand, the actual return of a portfolio in this model is simply the sum of individual returns, which are either 0 (unacceptable) or 1 (acceptable), weighted by the corresponding investments x_i, $i = 1, \ldots, n$. In other words, it is



(a) (b) (c)

Figure 2.7: The optimal investment (from [133]).

the proportion of investment allocated to acceptable solutions, the expected value and variance of which are precisely $\sum_{i=1}^{n} p_i x_i$ and $\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} x_i x_j$, respectively (refer to expressions (2.8a) and (2.8b)). If constant investment in each solution is assumed, then expected return becomes proportional to the expected number of solutions acceptable to a random DM. In contrast, the hypervolume indicator is proportional to the probability of there being at least one acceptable solution under the same conditions.

Yevseyeva *et al.* [133] observed how the optimal investment is distributed for a population whose individuals in the objective space are equally spaced on a linear front with -1 slope, and for a randomly generated population. These results are reproduced in Figure 2.7(a) and 2.7(b), respectively, where the area of the circle around a point is proportional to the investment in the solution that maps to that point. In Figure 2.7(a) it is possible to observe that the optimal investment seems to be the one where the investment is equally distributed. In Figure 2.7(b), strictly positive investment is only given to nondominated solutions and solutions in less crowded regions have higher fitness. Therefore, these figures indicate that the investment is split in such a way that good solutions receive strictly positive investment and diversity is preserved.

Moreover, an interesting fact is that this formulation allows constraints on the investment to be added easily. Figure 2.7(c) shows an example of the optimal portfolio when the maximum investment in one solution is $\frac{2}{n}$, where n is the total number of individuals. In this case, it is possible to observe that the optimal investment is such that the best solutions receive as much investment as possible, and the remaining investment is shared by some of the worst solutions. This behavior has been known in EAs as Disruptive Selection [101], but it is noteworthy that, in this case, it emerges from the very mathematical formulation of portfolio selection. This suggests that it may be possible to model other selection methods traditionally used in EAs in terms of a PSP.

Portfolio Optimization Selection Evolutionary Algorithm (POSEA)

Yevseyeva *et al.* [133] also proposed the Portfolio Optimization Selection Evolutionary Algorithm (POSEA), a $(\mu + \lambda)$ Evolutionary Algorithm that includes the Portfolio Selection formulation in the selection process. In the first iteration, after generating the initial population, the Sharpe ratio maximization problem is solved using a quadratic programming solver, and the investment is used as the fitness of each individual. SUS (Stochastic Universal Sampling) is used to select λ individuals for reproduction from the μ individuals in the population. For the environmental selection, the Sharpe ratio is maximized considering the $\mu + \lambda$ individuals and the λ individuals with worst fitness (investment) are chosen to die. Note that after the first iteration, the Sharpe ratio does not have to be recomputed for the remaining μ if all discarded individuals were assigned zero fitness. In that case, environmental selection and fitness assignment are performed in a single step. In fact, that can be ensured by adding an integer constraint such that a maximum of μ points are assigned strictly positive investment. However, such a constrained Sharpe-ratio maximization problem is no longer a convex quadratic programming problem.

Experiments performed by Yevseyeva *et al.* [133] on a multiobjective knapsack problem showed that viewing selection as a Portfolio Selection Problem is promising. In such experiments, only the mutation operator was used, and was applied to all new individuals. The results confirmed that using the proposed selection method based on Portfolio Selection, POSEA was able to find and maintain a good and diverse set of solutions.

Portfolio Selection vs Subset Selection

From a quality indicator point of view, environmental selection is in itself a subset selection problem, where μ out of $\mu + \lambda$ solutions have to be selected such that the indicator is maximized. This problem can be viewed as a particular case of portfolio selection with a constraint on the number of assets into which to invest, where all the selected solutions are assigned equal (strictly positive) investment (cardinality constraint).

When environmental selection is implemented just as a subset selection problem, fitness assignment has to be performed separately. Portfolio selection allows environmental selection and parental selection to be treated as a single problem.

2.4.6 Properties of Selection Methods

In $(\mu + \lambda)$ strategies, the population typically works as a bounded archive that selects the best μ solutions to keep. Environmental selection can then be viewed as what is called an *archiver*, a method to update the archive as new solutions are generated [36]. Ideally, dominated solutions should never be preferred over dominating ones, otherwise the quality of the population may degrade and convergence to the Pareto front might not be possible [147, 106]. For example, the Pareto-based selection methods used in NSGA-II and SPEA2 cannot guarantee that future populations/archives are not worse than past ones with respect to Pareto dominance, while selection methods based on the maximization of the hypervolume indicator, as in [98, 60], can [106]. Ideally, the implicit/explicit integration of preference information should provide a refinement of the Pareto-dominance towards a total order on solutions or on sets of solutions [87, 147].

In general, indicator-based selection methods are advantageous due to their properties that allow conclusions to be drawn about their ability to converge to the Pareto front, and the characterization of sets of solutions they give preference to. Among the main reasons for the popularity of the hypervolume indicator are its theoretical properties.

An overview of relevant properties of set-quality indicators can be found in [140]. Among those properties, parameter independence, sensitivity to scaling, monotonicity, and optimal μ-distributions, are considered in this work. While monotonicity properties allow some conclusions to be drawn about how indicator-optimal subsets relate to the Pareto front, the study of optimal μ-distributions (where $\mu = k$ in Problem 2.1) provides a more complete characterization of such optimal subsets, including how they depend on the indicator's parameters and/or on the scaling of the objectives.

Parameter independence

Set-quality indicators may include free parameters which affect how indicator values are computed and, thus, the corresponding preference structure. The hypervolume indicator, for example, has the coordinates of a reference point as its parameters. The location of this reference point provides some degree of control over how much preference is given to the presence of points on the boundaries of the Pareto front in a set, but the appropriate choice of a reference point depends on the number of objectives and on the shape of the Pareto front [4, 40, 91]. At least for fronts other than linear fronts, setting this reference point remains an open question.

Broadly speaking, indicator parameters are useful as long as they offer the decision maker an effective and well-understood means of expressing their preferences, but the more parameters there are, the more difficult it may be to translate given preferences into suitable parameter settings. One special case concerns parameters which do not actually influence the order which the quality indicator defines on the set of all subsets of the objective space, although they do affect indicator values. In that case, the quality indicator is said to be *independent* of those parameters, which may nevertheless be required.

Sensitivity to objective scaling

If the values taken by a set-quality indicator remain unchanged under coordinatewise strictly monotonic transformations of the objective space, then that indicator is said to be *scaling invariant* [140]. It follows naturally that indicators that do not depend on actual objective values, such as the cardinality indicator and the indicator defined as the fraction of the points in the (finite) Pareto front that are weakly dominated by the points in a given set, have this property.

A weaker form of insensitivity to scaling is called *scaling independence*, and consists in the order defined by a quality indicator being preserved under scaling transformations, possibly accompanied by corresponding transformations of the indicator's parameters. Both scale invariance and scale independence can be weakened by restricting the transformations considered to, e.g., affine, or even linear, transformations.

Monotonicity

Monotonicity properties formalize the empirical notion of agreement between indicator values and set dominance. Considering maximization of the indicator, without loss of generality, monotonicity of a set-quality indicator with respect to a set-dominance relation is defined as follows [140]:

Definition 2.15 (Monotonicity). A set-indicator I is *weakly monotonic* with respect to a set-dominance relation $\mathcal{R}$ (or weakly $\mathcal{R}$-monotonic, for short) if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A} \ \mathcal{R} \ \mathrm{B}$ implies $I(\mathrm{A}) \geq I(\mathrm{B})$. If $\mathrm{A} \ \mathcal{R} \ \mathrm{B}$ implies $I(\mathrm{A}) > I(\mathrm{B})$, the indicator is *strictly monotonic* with respect to $\mathcal{R}$ (or strictly $\mathcal{R}$-monotonic).

It is generally accepted that quality indicators used in EA environmental selection or in performance assessment should be weakly $\preceq$-monotonic at the very least, so that a set of points is never considered to be worse than any set it dominates, and that indifferent sets remain so with respect to the indicator. Nonetheless, as pointed out by Zitzler *et al.* [140], weak $\preceq$-monotonicity does not guarantee that all indicator-optimal subsets of up to a given size include points on the Pareto front. Indeed, even a quality indicator whereby all point sets are assigned the same value has this property. Still, $\preceq$-monotonicity does guarantee that at least one subset of the Pareto front is indicator-optimal, provided that an indicator-optimal subset of $f(\Omega)$ does exist (see [5]), as stated in the following Lemma [81]. The set of all nondominated points in a set $\mathrm{X} \subset \mathbb{R}^d$ is denoted by $\mathsf{nondominated}(\mathrm{X}) = \{q \in \mathrm{X} \mid \forall_{t \in \mathrm{X}}, \, t \leq q \Rightarrow q \leq t\}$.

Lemma 2.1. *Let* $\mathrm{S} \subset \mathbb{R}^d$ *be a non-empty set, let* $\mathrm{P} = \mathsf{nondominated}(\mathrm{S})$*, and let* I *be a weakly* $\preceq$*-monotonic set-quality indicator. For each integer* $k \geq 0$*, if an optimal subset of* S *with respect to* I *and* k *exists, then there is a subset* $\mathrm{A}_k^* \subseteq \mathrm{P}$ *of size* $|\mathrm{A}_k^*| = \min(k, |\mathrm{P}|)$ *that is also an optimal subset of* S *with respect to* I *given* k*, i.e.,*

$$I(\mathrm{A}_k^*) = \max_{\substack{\mathrm{B} \subseteq \mathrm{S} \\ |\mathrm{B}| \leq k}} I(\mathrm{B}) \, .$$

Proof. For $k = 0$, there is a single subset of the given size (the empty set), which is therefore indicator-optimal. For $k > 0$, this is proved by contradiction. Assume that there is no such subset A_k^*. Then, there must be an indicator-optimal subset $\mathrm{B} \subset \mathrm{S}$ of size up to k such that $|\mathrm{B} \cap \mathrm{P}| < \min(k, |\mathrm{P}|)$. Let $\mathrm{A} \subseteq \mathrm{P}$ be a point set constructed as follows. Beginning with $\mathrm{A} = \emptyset$, for each point in B, add to A a point in P that weakly dominates it. Then, continue adding to A points from P not yet in A until $|\mathrm{A}| = \min(k, |\mathrm{P}|)$. By this construction, $\mathrm{A} \preceq \mathrm{B}$, which implies $I(\mathrm{A}) \geq I(\mathrm{B})$ due to the weak $\preceq$-monotonicity of I. Therefore, A is also indicator-optimal, and a contradiction arises. □

Inclusion of points on the Pareto front in all indicator-optimal subsets up to a given size can still be guaranteed by requiring strict monotonicity with respect to a suitable non-weak set-dominance relation [81]. This is summarized in Table 2.1, but note that strict $\prec\!\!\prec$-monotonicity is not sufficient. Consider a set composed of two points, one of which strictly, but not strongly, dominates the other. A strictly $\prec\!\!\prec$-monotonic indicator may assign equal value to any subset of this set, even if it is also weakly $\preceq$-monotonic. So, the subset containing only the dominated solution can be indicator-optimal despite not including the only point on the Pareto front.

In contrast, a strictly $\prec\cdot$-monotonic indicator leads to all indicator-optimal subsets including at least one point on the Pareto front:

Lemma 2.2. *Let* $\mathrm{S} \subset \mathbb{R}^d$ *be a non-empty set, let* $\mathrm{P} = \mathsf{nondominated}(\mathrm{S})$*, let* I *be a strictly* $\prec\cdot$*-monotonic set-quality indicator, and let* k *be a positive integer. If* $\mathrm{A}_k^* \subset \mathrm{S}$ *is an optimal subset of* S *with respect to* I *given* k*, then* $|\mathrm{A}_k^* \cap \mathrm{P}| \geq 1$.

$\mathcal{R}$	$\lvert \mathrm{A}_k^* \cap \mathrm{P} \rvert$
$\prec\!\!\prec$	≥ 0
$\prec\cdot$	≥ 1
$\prec$	$\min(k, \lvert\mathrm{P}\rvert)$

Table 2.1: Consequences of strict indicator monotonicity with respect to set-dominance relations $\mathcal{R}$, where A_k^* denotes an indicator-optimal set of size up to k and P denotes the Pareto front.

Proof. Assume that there is a non-empty indicator-optimal subset $\mathrm{B} \subset \mathrm{S}$ such that $|\mathrm{B}| \leq k$ and $\mathrm{B} \cap \mathrm{P} = \emptyset$. Consequently, $\mathrm{P} \prec\cdot \mathrm{B}$. Let $\mathrm{A} \subseteq \mathrm{P}$ be a subset of points constructed by adding to the empty set, for each point in B, a point in P that strictly dominates it, and note that $1 \leq |\mathrm{A}| \leq k$. Since $\mathrm{A} \prec\cdot \mathrm{B}$ by construction, and the indicator is strictly $\prec\cdot$-monotonic, it follows that $I(\mathrm{A}) > I(\mathrm{B})$, which contradicts the assumption that B is indicator-optimal. □

Finally, strict $\prec$-monotonicity guarantees that all indicator-optimal subsets either contain or are maximum-size subsets of the Pareto front:

Lemma 2.3. *Let $\mathrm{S} \subset \mathbb{R}^d$ be a non-empty set, let $\mathrm{P} = \textsf{nondominated}(\mathrm{S})$, let I be a strictly $\prec$-monotonic set-quality indicator, and let k be a positive integer. If $\mathrm{A}_k^* \subset \mathrm{S}$ is optimal with respect to I given k, then $|\mathrm{A}_k^* \cap \mathrm{P}| = \min(k, |\mathrm{P}|)$.*

Proof. Assume that there is a non-empty indicator-optimal subset $\mathrm{B} \subset \mathrm{S}$ such that $|\mathrm{B}| \leq k$ and $|\mathrm{B} \cap \mathrm{P}| < \min(k, |\mathrm{P}|)$. Consequently, $\mathrm{P} \prec \mathrm{B}$. Let $\mathrm{A} \subseteq \mathrm{P}$ be a subset of points constructed by adding to the empty set, for each point in B, a point in P that weakly dominates it, and then adding any other points from P until $|\mathrm{A}| = \min(k, |\mathrm{P}|)$. Since $\mathrm{A} \prec \mathrm{B}$ by construction, and the indicator is strictly $\prec$-monotonic, it follows that $I(\mathrm{A}) > I(\mathrm{B})$, which contradicts the assumption that B is indicator-optimal. □

Ideally, quality indicators should be strictly $\prec$-monotonic [148, 140], so that any slight improvement of a point set over another is always reflected by a higher indicator value being assigned to the former, and that the indicator is maximized by sets containing the whole Pareto front. The hypervolume indicator and some of its variants [138, 70] are the only indicators known to possess such a property, provided (at least) that the reference point is strongly dominated by all points in the parent set. However, this comes at a potentially high computational cost, as the hypervolume indicator cannot be computed exactly in polynomial time in the number of objectives unless $\mathbf{P} = \mathbf{NP}$ [32]. Most other popular set-quality indicators, such as the ϵ-indicator [140] and those in the R-indicator family [140, 43] are known to be weakly $\preceq$-monotonic and, at best, strictly $\prec\!\!\prec$-monotonic, as is the case with the ϵ-indicator [148]. A notable exception is the Averaged Hausdorff Distance indicator [117], which is known not to be weakly $\preceq$-monotonic. However, a more detailed analysis suggests that the *degree* to which the $\preceq$-monotonicity condition is violated depends on the maximum subset size and on the shape of the Pareto front, at least in the two-objective case [116].

Optimal μ-distributions

Beyond whether or not indicator-optimal subsets include points on the Pareto front, a more complete characterization of such optimal subsets has been sought through the study of so-called *optimal μ-distributions*, where μ refers to the maximum subset size. This entails determining actual indicator-optimal subsets for given (families of) Pareto fronts or, at least, some aspect of the location of the corresponding points in objective space. For indicators that are strictly $\prec$-monotonic, the analysis can be somewhat simplified by noting that all points in an indicator-optimal subset of sufficiently small size must lie on the Pareto front (Lemma 2.3). Otherwise, it may still be appropriate to restrict the analysis to points on the Pareto front if the indicator is weakly $\preceq$-monotonic (Lemma 2.1), but failing that all point subsets must be considered.

Most studies to date only consider the two-objective case. Concerning the hypervolume indicator such studies showed that, for example, for a linear front in the two objective case, there is a unique optimal distribution where all points are uniformly spaced. In the case of the $R2$-indicator with a weighted Tchebycheff utility function, using μ weight vectors uniformly distributed in weight space also leads to unique, uniform distributions on linear fronts [43].

Given the absence of more general theoretical results, numerical approaches have been used to approximate optimal μ-distributions and the corresponding (maximum) indicator values on two-objective [38, 72], and even three-objective [72], continuous benchmark problems.

2.5 Concluding Remarks

Selection plays an import role in EMOAs. It is through selection that preference information, whether explicit or implicit, influences the search process. The theoretical characterization of selection methods allows us to better understand how EMOAs behave and, consequently, to better foresee their outcomes, and to make better use of EMOAs. Indicator-based selection methods look at selection as a subset selection problem and can be studied as such. Hypervolume-based selection methods are among those with better theoretical properties, and thus among the state-of-the-art. For this reason, they will be further studied in the next two chapters (Chapters 3 and Chapters 4).

Portfolio selection-based methods provide a promising perspective on selection and fitness assignment. It remains to be shown whether this more general perspective can be at least as theoretically supported as, and be competitive in practice with, the existing selection methods based on subset selection, as well as which additional advantages it may bring to EAs in general. This new perspective motivates the work in Chapter 5.

Chapter 3

Hypervolume Indicator

The hypervolume set-quality indicator maps a point set in $\mathbb{R}^d$ to the measure of the region dominated by that set and bounded above by a given reference point, also in $\mathbb{R}^d$. It was first referred to as the "size of the space covered" [144, 145], and as "size of the dominated space" [137]. Alternative designations have also been used, such as $\mathcal{S}$-metric [145, 17] and "Lebesgue measure" [62]. Different definitions of this indicator have been proposed. For example, it has been defined based on the union of polytopes [145] and, more generally based on the (integration of the) attainment function [138, 77]. The problem of computing the hypervolume indicator is known to be a special case of Klee's Measure Problem (KMP) [14], which is the problem of measuring the region resulting from the union of axis-parallel boxes. The hypervolume indicator is, in fact, a special case of KMP on unit cubes, and of KMP on grounded boxes [134]. See [30] for a review on KMP's special cases and their relation to one another.

The hypervolume indicator was first proposed as a method for assessing multiobjective optimization algorithms [145]. It evaluates the optimizer outcome by simultaneously taking into account the proximity of the points to the Pareto front, diversity, and spread, which are the features most commonly used for evaluation when DM preferences are not available [139]. The indicator's unique properties quickly led to its integration in EMOAs, as a bounding method for archives [98], as an environmental selection method [60], as a ranking method [90], and as fitness assignment method [141, 10, 147]. The integration of preferences in the indicator [138, 3, 41] has also been the subject of discussion, and so has the integration of diversity in the decision space [125]. Currently, the hypervolume indicator is one of the indicators used in the Black-Box Optimization Benchmarking (BBOB) tool [42] to continuously evaluate the external archive containing all nondominated solutions EMOAs generate during their execution.

The merits of the hypervolume indicator for performance assessment are well recognized, and so are the benefits of incorporating it in EMOAs. However, its main drawback lies in its computational cost. This is particularly relevant as hypervolume-based EMOAs and benchmarking tools such as BBOB depend heavily on its computation. This imposes strong limitations on the number of objectives considered and/or on EMOA parameters such as the number of generations and number of offspring. In order to overcome such a limitation, approximation algorithms [10] have been proposed, as well as objective reduction methods [44].

In the following sections, the theoretical advantages and the computational issues surrounding the hypervolume indicator will be discussed in more detail, mostly in the context of EMOAs. In Section 3.1, the hypervolume indicator and some related problems are formally defined, and its properties are reviewed in Section 3.2. In Section 3.3, hypervolume-based algorithms for environmental selection are discussed. A review of the state-of-the-art algorithms for hypervolume-related problems is provided in Section 3.4. The data sets commonly used for evaluating the performance of hypervolume-related algorithms are presented in Section 3.5, as well as some new data sets that will be used in Chapter 4. Concluding remarks are drawn in Section 3.6.

3.1 Hypervolume-related Problems

3.1.1 Notation

Spaces (x, y)-, (x, y, z)- and (x, y, z, w)-spaces will be referred to as, 2-, 3- and 4-dimensional spaces or, for brevity, as 2D, 3D and 4D spaces, respectively.

Problem size The lower-case letter n is used for the problem size, which is typically the size of the input set.

Number of dimensions The lower-case letter d is used to represent the number of dimensions considered.

Points and sets Points are represented by lower-case letters (in italics) and sets by Roman capital letters. For example, $p, q \in \mathbb{R}^d$ and $\mathrm{X}, \mathrm{S} \subset \mathbb{R}^d$.

Coordinates (for $d \leq 4$) Letters x, y, z and w in subscript denote the coordinates of a point in an (x, y, z, w)-space. This notation is used only for spaces up to 4 dimensions. For example, if $p \in \mathbb{R}^3$ then $p = (p_x, p_y, p_z)$.

Coordinates (general case $d \geq 2$) In general d-dimensional spaces, an index in subscript is used to identify the coordinate. For example, if $p \in \mathbb{R}^d$ then $p = (p_1, p_2, \ldots, p_d)$, where p_i denotes the i^{th} coordinate of p, $i \in \{1, \ldots, d\}$.

Enumeration Numbers in superscript are used to enumerate points or sets, e.g., $p^1, p^2, p^3 \in \mathbb{R}^d$ and $\mathrm{S}^1, \mathrm{S}^2 \subset \mathbb{R}^d$.

Projections Projection onto $(d-1)$-space by omission of the last coordinate is denoted by an asterisk. For example, given the point set $\mathrm{X} = \{p, q\} \subset \mathbb{R}^3$, p^* and X^* denote the projection of the point p and of the point set X on the (x, y)-plane, respectively, i.e., $p^* = (p_x, p_y)$ and $\mathrm{X}^* = \{(p_x, p_y), (q_x, q_y)\}$.

3.1.2 Definitions

The hypervolume indicator [98, 144] is formally defined as follows:

Definition 3.1 (Hypervolume Indicator). Given a point set $\mathrm{S} \subset \mathbb{R}^d$ and a reference point $r \in \mathbb{R}^d$, the hypervolume indicator of S is the measure of the region weakly dominated by S and bounded above by r, i.e.:

$$H(\mathrm{S}) = \Lambda\left(\bigcup_{\substack{p\in \mathrm{S}\\ p\leq r}} [p, r]\right)$$

where $[p, r] = \{q \in \mathbb{R}^d \mid p \leq q \textbf{ and } q \leq r\}$ and $\Lambda(\cdot)$ denotes the Lebesgue measure. Alternatively:

$$H(\mathrm{S}) = \Lambda(\{q \in \mathbb{R}^d \mid \exists p \in \mathrm{S} : p \leq q \textbf{ and } q \leq r\})$$

Since a fixed reference point, r, is assumed throughout this thesis, it is omitted as an argument of $H(\cdot)$ function. Figure 3.1(a) shows a two-dimensional example of the hypervolume (an area) and Figure 3.2(a) shows a three-dimensional example (a volume).

The hypervolume contribution of a point set to some reference point set [36, 39] is formally defined based on the definition of hypervolume indicator:

Definition 3.2 (Hypervolume Contribution of a Point Set). Given two point sets $\mathrm{X}, \mathrm{S} \subset \mathbb{R}^d$, and a reference point $r \in \mathbb{R}^d$, the (hypervolume) contribution of X to S is:

$$H(\mathrm{X}, \mathrm{S}) = H(\mathrm{X} \cup \mathrm{S}) - H(\mathrm{S} \setminus \mathrm{X})$$

Note that if $\mathrm{X} \cap \mathrm{S} = \emptyset$ then the contribution of X to S is simply $H(\mathrm{X}, \mathrm{S}) = H(\mathrm{X} \cup \mathrm{S}) - H(\mathrm{S})$. See Figure 3.1(b) for a two-dimensional example. The particular case of $|\mathrm{X}| = 1$ is used more frequently and is defined as in [36]:

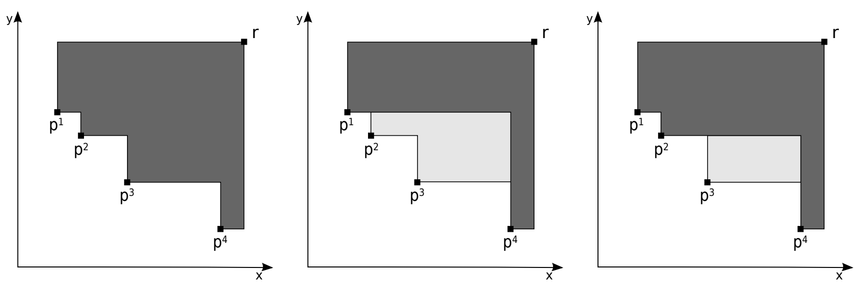
Definition 3.3 (Hypervolume Contribution). The hypervolume contribution of a point $p \in \mathbb{R}^d$ to a set $\mathrm{S} \subset \mathbb{R}^d$ is:

$$H(p, \mathrm{S}) = H(\mathrm{S} \cup \{p\}) - H(\mathrm{S} \setminus \{p\})$$

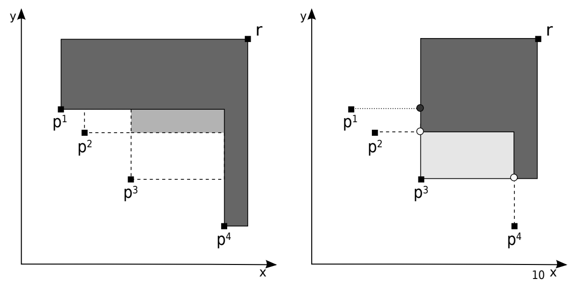
The hypervolume *contribution* of a point is sometimes referred in the literature as the *incremental* hypervolume or the *exclusive* hypervolume [128]. Moreover, the contribution of a point p to the empty set is sometimes called the *inclusive* hypervolume [128]. See Figures 3.1(c) and 3.2(b) for two-dimensional and three-dimensional examples of a hypervolume contribution, respectively.

As pointed out in [36], the above Definition 3.3 is consistent with the case where $p \in \mathrm{S}$, and the contribution is the hypervolume lost when p is removed from S, as well as with the case where $p \notin \mathrm{S}$, and the contribution of p is the hypervolume gained when adding p to S. While this is certainly convenient, it does not reflect the fact that the hypervolume gained by "adding" a point p to a set already including it is zero. However, this last situation can be handled easily as a special case by checking whether S includes p before applying the definition.

In some cases, such as when determining the decrease in the contribution of a given point $p \in \mathbb{R}^d$ to a set $\mathrm{S} \subset \mathbb{R}^d$ due to the addition of another point $q \in \mathbb{R}^d$ to S, it is also useful to consider the contribution dominated simultaneously and exclusively by two points.

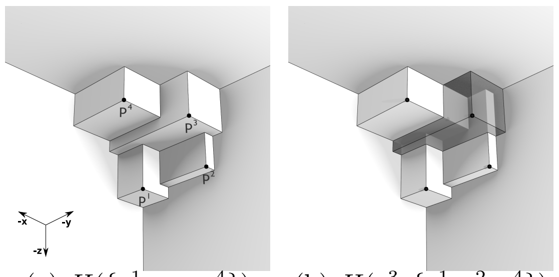


(a) $H(\{p^1,\dots,p^4\})$ (b) $H(\{p^2,p^3\},\{p^1,p^4\})$ (c) $H(p^3,\{p^1,p^2,p^4\})$



(d) $H(p^2,p^3,\{p^1,p^4\})$ (e) Delimiters of p^3

Figure 3.1: Examples in two-dimensions: (a) hypervolume indicator (dark gray region), (b) hypervolume contribution of a point set (light gray region), (c) hypervolume contribution of a point (light gray region), (d) joint hypervolume contribution (mid gray region), and (e) delimiters of p^3 (p^2 and p^4), as well as dominated point ($p^3 \vee p^1$), represented by a filled circle, and nondominated points ($p^3 \vee p^2$) and ($p^3 \vee p^4$), elements of J, represented by hollow circles (see text for more details).



(a) $H(\{p^1,\dots,p^4\})$ (b) $H(p^3,\{p^1,p^2,p^4\})$

Figure 3.2: Three-dimensional examples: (a) hypervolume indicator (opaque volume), (b) hypervolume contribution (transparent volume).

Definition 3.4 (Joint Hypervolume Contribution). The joint hypervolume contribution of $p,q \in \mathbb{R}^d$ to $\mathrm{S} \subset \mathbb{R}^d$ is:

$$H(p,q,\mathrm{S}) = H((\mathrm{S} \setminus \{p,q\}) \cup \{p \vee q\}) - H(\mathrm{S} \setminus \{p,q\})$$

where $\vee$ denotes the *join*, or component-wise maximum between two points.

Figures 3.1(d) shows an example for the two-dimensional case and Figure 3.3 for the three-dimensional case. In the $d = 3$ example, Figure 3.3(a) shows the individual contribution of p^3 and p^4 to $\mathrm{S} = \{p^1, p^2\}$, which partially overlap. This partially overlapped volume is the joint contribution (represented in transparent gray in Figure 3.3(d)). This joint contribution can also be interpreted as the region of the

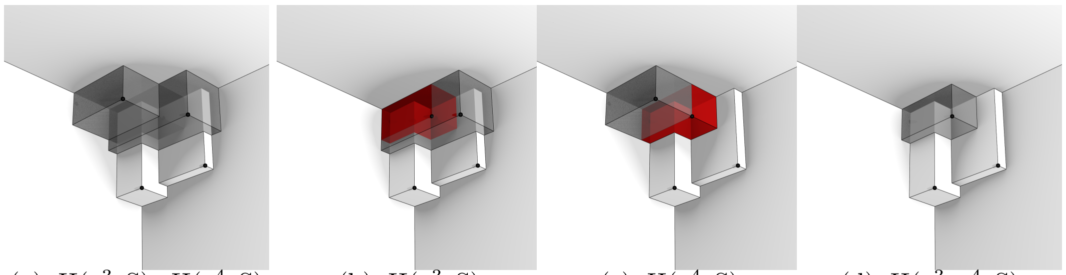

(a) $H(p^3, \mathrm{S})$, $H(p^4, \mathrm{S})$ (b) $H(p^3, \mathrm{S})$ (c) $H(p^4, \mathrm{S})$ (d) $H(p^3, p^4, \mathrm{S})$

Figure 3.3: Three-dimensional example, based on Figure 3.2(a), of the joint contribution of p^3 and p^4 to $\mathrm{S} = \{p^1, p^2\}$, i.e., $H(p^3, p^4, \{p^1, p^2\})$. Contributions are shown in transparent gray and red. Transparent red highlights the part of a contribution also dominated by the omitted point.

contribution of p^3 to S that is also dominated by p^4 (the transparent red region in Figure 3.3(b)). Analogously, it can be interpreted as the region of the contribution of p^4 to S that is also dominated by p^3 (the transparent red region in Figure 3.3(c)).

Moreover, the contribution of a point p to a set S is bounded above by certain points $q \in \mathrm{S}$ that shall be referred to as *delimiters*, and are defined as follows:

Definition 3.5 (Delimiter). Given a point set $\mathrm{S} \subset \mathbb{R}^d$ and a point $p \in \mathbb{R}^d$, let $\mathrm{J} = \mathsf{nondominated}(\{(p \vee q) \mid q \in \mathrm{S} \setminus \{p\}\})$, where $\mathsf{nondominated}(\mathrm{X}) = \{s \in \mathrm{X} \mid \forall_{t \in \mathrm{X}}, t \leq s \Rightarrow s \leq t\}$ denotes the set of nondominated points in X. Then, $q \in \mathrm{S}$ is called a *(weak) delimiter* of the contribution of p to S iff $(p \vee q) \in \mathrm{J}$. If, in addition, $H(p, q, \mathrm{S}) > 0$, then q is also a *strong delimiter* of the contribution of p to S.

Note that J is the smallest set of points weakly dominated by p that delimits its contribution to S, that is, $H(p, \mathrm{S}) = H(p, \mathrm{J})$. Consequently, all $q \in \mathrm{J}$ are such that $H(p, q, \mathrm{J}) > 0$, and J is such that $H(p, \mathrm{S}) = H(\{p\}) - H(\mathrm{J})$. Figure 3.1(e) shows an example where the contribution of p^3 is delimited only by p^2 and p^4, where both p^2 and p^4 are strong delimiters.

Non-strong delimiters can only exist when S contains points with repeated coordinates. In the example of Figure 3.4(a), p^4 and p^5 are strong delimiters while p^2 and p^3 are not strong but only weak delimiters. This means that, in practice, only one point in such a group of delimiters is needed to bound the contribution of p. If one of them is deleted, then the contribution of p remains unchanged, whereas it increases if all are deleted.

The following extension to the notion of delimiter will also be needed in this work:

Definition 3.6 (Outer Delimiter). Given a point set $\mathrm{S} \subset \mathbb{R}^d$ and a point $p \in \mathbb{R}^d$, $q \in \mathrm{S}$ is called an *outer delimiter* of the contribution of p to S if it is a delimiter of the contribution of p to $\{s \in \mathrm{S} \mid p \not\leq s\}$. A delimiter, q, of the contribution of p to S is called an *inner delimiter* if it is not an outer delimiter, i.e., if $p \leq q$.

In general, outer delimiters may not be actual, or *proper*, delimiters in the sense of Definition 3.5, in particular, when p shares a coordinate with a point in S. In the example of Figure 3.4(b), points p^3, p^4, p^5 are the (proper) delimiters of the

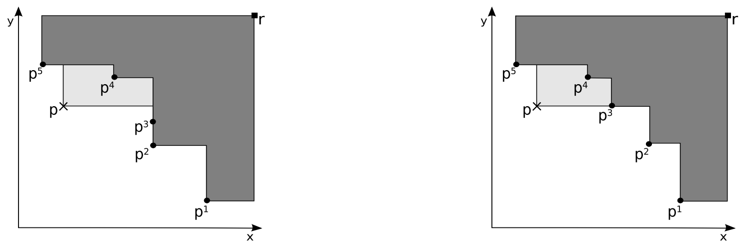


Figure 3.4: Examples of the delimiters of the contribution of p to a point set containing points with repeated coordinates.

contribution of p to S, of which p^3 and p^4 are inner delimiters. There are two outer delimiters, p^2 and p^5. Point p^2 is not a proper delimiter of the contribution of p to S because $(p \vee p^3) = p^3 < (p \vee p^2)$.

3.1.3 Problems

Many computational problems related to the hypervolume indicator can be found in the literature. The following problems are needed in the context of this work, and are based on definitions given in Emmerich and Fonseca [61]. Recall that the reference point is considered to be a constant.

Problem 3.1 (HYPERVOLUME)**.** Given an n-point set $\mathrm{S} \subset \mathbb{R}^d$ and a reference point $r \in \mathbb{R}^d$, compute the hypervolume indicator of S, i.e., $H(\mathrm{S})$.

Problem 3.2 (ONECONTRIBUTION)**.** Given an n-point set $\mathrm{S} \subset \mathbb{R}^d$, a reference point $r \in \mathbb{R}^d$ and a point $p \in \mathbb{R}^d$, compute the hypervolume contribution of p to S, i.e., $H(p, \mathrm{S})$.

Problem 3.3 (ALLCONTRIBUTIONS)**.** Given an n-point set $\mathrm{S} \subset \mathbb{R}^d$ and a reference point $r \in \mathbb{R}^d$, compute the hypervolume contributions $H(p, \mathrm{S})$ of all points $p \in \mathrm{S}$ to S.

Problem 3.4 (ALLCONTRIBUTIONS2)**.** Given an n-point set $\mathrm{S} \subset \mathbb{R}^d$, an m-point set $\mathrm{R} \subset \mathbb{R}^d$ such that $\mathrm{S} \cap \mathrm{R} = \emptyset$ and a reference point $r \in \mathbb{R}^d$, compute the hypervolume contributions $H(p, \mathrm{R})$ of all points $p \in \mathrm{S}$ to R.

Note that, by definition, the contributions of two points $p, q \in \mathrm{S}$ to S never overlap in Problem 3.3 while in Problem 3.4, the contributions of $p, q \in \mathrm{S}$ to a point set R may overlap. For example, in Figure 3.1(d), if $\mathrm{R} = \{p^1, p^4\}$ and $\mathrm{S} = \{p^2, p^3\}$, the contribution of p^2 to R includes the region in lighter gray, and so does the contribution of p^3 to R, in addition to the respective (exclusively) dominated white regions. However, in Problem 3.3, if $\mathrm{S} = \{p^1, \ldots, p^4\}$ then their contributions reduce just to the respective white regions, i.e., do not include the joint contribution.

Problem 3.5 (LEASTCONTRIBUTOR)**.** Given an n-point set $\mathrm{S} \subset \mathbb{R}^d$ and a reference point $r \in \mathbb{R}^d$, find a point $p \in \mathrm{S}$ with minimal hypervolume contribution to S.

Sometimes, the above problems are computed for a sequence of sets that differ in a single point from the previous one, either by adding a point to (Incremental case) or by removing a point from (Decremental case) the previous set.

Problem 3.6 (UPDATEHYPERVOLUME)**.** Given an n-point set $\mathrm{S} \subset \mathbb{R}^d$, the reference point $r \in \mathbb{R}^d$, the value of $H(\mathrm{S})$, and a point $p \in \mathbb{R}^d$, compute:

- **Incremental:** $H(\mathrm{S} \cup \{p\}) = H(\mathrm{S}) + H(p, \mathrm{S})$, where $p \notin \mathrm{S}$.
- **Decremental:** $H(\mathrm{S} \setminus \{p\}) = H(\mathrm{S}) - H(p, \mathrm{S})$, where $p \in \mathrm{S}$.

Problem 3.7 (UPDATEALLCONTRIBUTIONS)**.** Given an n-point set $\mathrm{S} \subset \mathbb{R}^d$, a reference point $r \in \mathbb{R}^d$, the value of $H(q, \mathrm{S})$ for every $q \in \mathrm{S}$, and a point $p \in \mathbb{R}^d$:

- **Incremental:** Compute $H(q, \mathrm{S} \cup \{p\}) = H(q, \mathrm{S}) - H(p, q, \mathrm{S})$ for all $q \in \mathrm{S}$, and also $H(p, \mathrm{S})$, where $p \notin \mathrm{S}$.
- **Decremental:** Compute $H(q, \mathrm{S} \setminus \{p\}) = H(q, \mathrm{S}) + H(p, q, \mathrm{S})$ for all $q \in \mathrm{S} \setminus \{p\}$, where $p \in \mathrm{S}$.

Problem 3.8 (UPDATEALLCONTRIBUTIONS2)**.** Given an n-point set $\mathrm{S} \subset \mathbb{R}^d$, an m-point set $\mathrm{R} \subset \mathbb{R}^d$, a reference point $r \in \mathbb{R}^d$, the value of $H(q, \mathrm{R})$ for every $q \in \mathrm{S}$, and a point $p \in \mathbb{R}^d$:

- **Incremental:** Compute $H(q, \mathrm{R} \cup \{p\}) = H(q, \mathrm{R}) - H(p, q, \mathrm{R})$ for all $q \in \mathrm{S}$, where $p \notin \mathrm{R} \cup \mathrm{S}$.
- **Decremental:** Compute $H(q, \mathrm{R} \setminus \{p\}) = H(q, \mathrm{R}) + H(p, q, \mathrm{R})$ for all $q \in \mathrm{S}$, where $p \in \mathrm{R}$ and $p \notin \mathrm{S}$.

Finally, based on the definition by Bader and Zitzler [10], the Hypervolume Subset Selection Problem (HSSP)[1] is formally defined here as:

Problem 3.9 (HSSP)**.** Given a n-point set $\mathrm{S} \subset \mathbb{R}^d$ and an integer $k \in \{0, 1, \ldots, n\}$, find a subset $\mathrm{A} \subseteq \mathrm{S}$ such that $|\mathrm{A}| \leq k$ and:

$$H(\mathrm{A}) = \max_{\substack{\mathrm{B} \subseteq \mathrm{S} \\ |\mathrm{B}| \leq k}} H(\mathrm{B})$$

The complement problem of the HSSP is defined as:

Problem 3.10 (HSSPCOMPLEMENT)**.** Given a n-point set $\mathrm{S} \subset \mathbb{R}^d$ and an integer $k \in \{0, 1, \ldots, n\}$, find a subset $\mathrm{C} \subseteq \mathrm{S}$ such that $|\mathrm{C}| \geq (n - k)$ and:

$$H(\mathrm{C}, \mathrm{S}) = \min_{\substack{\mathrm{B} \subseteq \mathrm{S} \\ |\mathrm{B}| \geq (n-k)}} H(\mathrm{B}, \mathrm{S})$$

If $\mathrm{A} \subseteq \mathrm{S}$ is a solution to the HSSP given k and S, then $\mathrm{S} \setminus \mathrm{A}$ is a solution to the HSSPCOMPLEMENT, and vice-versa.

Note that, in the above problems, S is usually a nondominated point set, even though this is not mandatory. Any dominated point $q \in \mathrm{S}$ has a null contribution to S. However, if q is dominated by a single point $p \in \mathrm{S}$, then the contribution of p to S will be lower than what it would be if $q \notin \mathrm{S}$. Moreover, the incremental scenarios of Problems 3.6 to 3.8 explicitly require that $p \notin \mathrm{S}$ because the adopted definition of hypervolume contribution does not handle adding a point to a set in which it is already included, as discussed before, nor does it consider the multiset that would result from such an operation. If such cases become relevant, the hypervolume contribution of repeated points in a multiset should be considered to be zero.

[1]Note that, the HSSP has also been defined in [3] as the subset selection problem with respect to the Weighted Hypervolume Indicator of which the hypervolume indicator is a special case.

Relation between problems

Most of the problems listed above are not expected to be efficiently solved for an arbitrary number of dimensions d. For example, problems HYPERVOLUME [32] and ONECONTRIBUTION [33] are known to be **#P**-hard. Even deciding if a point is the least contributor is **#P**-hard [33]. Moreover, HSSP was recently shown to be **NP**-hard [31] for $d \geq 3$. Although these are not encouraging results for an arbitrary d, this does not mean that efficient algorithms to compute the hypervolume-related problems exactly, or to approximate the HSSP cannot be developed for a fixed and small d. To develop such efficient algorithms, it is important to understand how the various hypervolume problems relate to each other or if they arise as subproblems.

All problems above (Problems 3.1 to 3.10) are intrinsically related. For most of them, it is possible to solve each one by solving one or more instances of the others. Consequently, state-of-the-art algorithms frequently exploit these relations.

It is clear from Definition 3.3 that any algorithm that computes HYPERVOLUME (Problem 3.1) can also be used to compute ONECONTRIBUTION (Problem 3.2). Moreover, the UPDATEHYPERVOLUME problem (Problem 3.6) can be solved by computing either HYPERVOLUME given $\mathrm{S} \cup \{p\}$ or ONECONTRIBUTION given S and p. On the other hand, the HYPERVOLUME problem can be computed by solving a sequence of UPDATEHYPERVOLUME problems as new points are added to a set. For example, consider $\mathrm{S} = \{p^1, p^2, p^3\}$, the hypervolume $H(\mathrm{S})$ can be computed as the sum $H(p^1, \{\}) + H(p^2, \{p^1\}) + H(p^3, \{p^1, p^2\})$, more generally:

$$H(\mathrm{S}) = \sum_{i=1}^{n} H(p^i, \{p^1, \ldots, p^{i-1}\}) = H(p^1, \{\}) + H(p^2, \{p^1\}) + \ldots + H(p^n, \{p^1, \ldots, p^{n-1}\})$$

where $\mathrm{S} = \{p^1, \ldots, p^n\}$. In fact, when such points are sorted according to dimension d, then the sequence of subproblems become $(d-1)$-dimensional problems, as exploited by the dimension-sweep approach (see Section 3.4.1). Dedicated algorithms to solve UPDATEHYPERVOLUME (and the other update problems) can take advantage of previous calculations and data structures to avoid redundant (re)computations and consequently, to save time.

It should also be clear that any algorithm that computes ONECONTRIBUTION can be used to compute ALLCONTRIBUTIONS (Problem 3.3), and vice-versa. Algorithms to solve ALLCONTRIBUTIONS can also be used to solve LEASTCONTRIBUTOR (Problem 3.5), by computing all contributions and then selecting a point with minimal contribution, although it is not strictly required to know all contributions to find the least contributor (see IHSO [B4] algorithm in Section 3.4.3). In the absence of dedicated algorithms, UPDATEALLCONTRIBUTIONS (Problem 3.7) can be solved by recomputing all contributions (ALLCONTRIBUTIONS).

Analogously to Problem 3.3, algorithms to compute ONECONTRIBUTION can also be used to compute ALLCONTRIBUTIONS2 (Problem 3.4) problem. Moreover, UPDATEALLCONTRIBUTIONS2 (Problem 3.8) can also be solved by recomputing the contributions (ALLCONTRIBUTIONS2). Despite the similarities, ALLCONTRIBUTIONS and ALLCONTRIBUTIONS2 are distinct to the point that one cannot be straightforwardly used to solve the other. The same observation applies to the corresponding update problems (UPDATEALLCONTRIBUTIONS and UPDATEALLCONTRIBUTIONS2).

The definition of HSSP suggests that it can be solved by enumerating all subsets of size k and computing the hypervolume indicator of each one. Similarly, HSSPComplement can be computed in an analogous way. However, this is obviously not practical as the time required would quickly be unacceptable unless n is sufficiently small and k is either very small or close to n. Moreover, recall that an optimal solution to the HSSP can be obtained from an optimal solution to HSSPComplement and vice-versa. For the particular case of $k = n - 1$, the LeastContributor problem provides the solution to the HSSP by definition.

Finally, it is important to have in mind that the way problems are solved has effect on the precision of the calculations. As pointed out by Russo and Francisco [119], computing the contribution of a point as the subtraction of two large hypervolumes raises precision problems. It is thus recommended to avoid subtracting hypervolumes as much as possible and, ideally, performing subtractions only between coordinates. For example, regarding numerical stability, using a specific algorithm to compute OneContribution may be preferable to using an algorithm for Hypervolume to compute the contribution based on a subtraction.

Section 3.4 gives more details on the relation between hypervolume-based problems by explaining the existing algorithms and the techniques used by them.

3.2 Properties

As an indicator that imposes a total order among point sets, the hypervolume indicator is biased towards some type of distribution of the points on a front [138]. Understanding that bias by studying its properties allows to better understand the underlying assumptions on DM preferences (see Subsection 2.4.6 for a description of the properties here discussed).

The hypervolume indicator is well acknowledged by its properties. It is strictly $\prec$-monotonic and it is scaling independent [137, 97]. This implies that it is maximal for the Pareto front and that scaling objectives does not affect the order imposed on point sets (see Subsection 2.4.6).

Concerning optimal μ-distributions in two dimensions, the exact location of the points in an optimal subset of a given size μ is known only for continuous linear fronts [4]. In this case, there is a unique optimal subset where all points are on the Pareto front and uniformly spaced between two outer points, the position of which depends both on the reference point and on the two extreme points of the Pareto front [40]. General fronts have also been studied, but only in terms of point density on the Pareto front when the number of points μ tends to infinity [4]. Unfortunately, the results available for the two-objective case do not generalize easily to three objectives, and not much is known about optimal μ-distributions in this case. The main results concern whether there is a setting of the reference point guaranteeing the inclusion of a front's extreme points in the optimal μ-distribution [2, 122] and the derivation of the optimal μ-distributions for the special case of Pareto fronts consisting of a line segment embedded in a three-objective space [122].

Very recently, Ulrich and Thiele [126] showed that the hypervolume indicator is a submodular function, i.e., given a decision space $\mathcal{X}$ and a function $z : 2^{\mathcal{X}} \to \mathbb{R}$, z is submodular if:

$$\forall \mathrm{A}, \mathrm{B} \subseteq \mathcal{X}, \;\; z(\mathrm{A}) + z(\mathrm{B}) \geq z(\mathrm{A} \cup \mathrm{B}) + z(\mathrm{A} \cap \mathrm{B}) \tag{3.1}$$

Submodularity is an important property as it relates to convexity in combinatorial optimization [107]. Additionally, a submodular function z is non-decreasing (or monotone) if:

$$\forall \mathrm{A} \subseteq \mathrm{B} \subseteq \mathcal{X},\;\; z(\mathrm{A}) \leq z(\mathrm{B}) \tag{3.2}$$

The hypervolume indicator is a non-decreasing submodular function [126]. See [111] for alternative equivalent definitions of (non-decreasing) submodular functions and examples. Because the Hypervolume Subset Selection Problem (HSSP) consists of maximizing a submodular function subject to a cardinality constraint [71], the approximation of HSSP by means of a (incremental) greedy approach has an approximation guarantee [111]. Hence, the subset obtained by selecting k points from S one at a time so as to maximize the hypervolume gained at each step is a $(1-1/e)$-approximation to the hypervolume of an optimal subset, i.e., the ratio between the greedy solution and the optimal solution is greater than or equal to $(1-1/e) \simeq 0.63$.[2]

3.3 Hypervolume-based Archiving Algorithms

The population in $(\mu+\lambda)$-selection EAs works as a bounded archive of μ solutions where the method used for environmental selection can be viewed as an Archiving Algorithm (AA). Such a $(\mu+\lambda)$-archiving algorithm is used at every generation to select a set of μ solutions, P′, out of the μ solutions of the parent population, P, plus the new λ offspring, Q, i.e., $\mathrm{P}' \subseteq \mathrm{P} \cup \mathrm{Q}$. Hypervolume-based EMOAs aim at finding a μ-sized subset of the Pareto front that maximizes the hypervolume indicator, i.e., finding the optimal solution for the HSSP (see Problem 3.9) given $k = \mu$ and considering the feasible set of solutions for a problem. For that purpose, they use Hypervolume-based AAs which can be described as $(\mu+\lambda)$-AAs that use a hypervolume-related measure to select P′, for example, by selecting the μ-sized subset of $\mathrm{P}\cup\mathrm{Q}$ with maximum hypervolume indicator value, or by sequentially selecting a point based on its hypervolume contribution. The first known Hypervolume-based AA was a $(\mu+1)$ AA [98] that selects μ solutions to retain by greedily discarding the least contributor which, in such a setting $(\lambda = 1)$, allows for optimal selection regarding the hypervolume indicator. Such an AA is also used in the well-known SMS-EMOA [16].

Hypervolume-based AAs have to decide which solutions to keep based on the current population and offspring, i.e., without the knowledge of future offspring or past (discarded) solutions. Therefore, in general, they are not able to retain the optimal subset (the optimal μ-distribution) even though they may have seen all solutions in the optimal subsets. Consequently, the guarantees of convergence of such archiving algorithm, and consequently of the EMOAs using them, have been a topic of interest. This section is based on the article by Bringmann and Friedrich [36] that reviews and extends the convergence theory of hypervolume-based AAs.

3.3.1 Classes

There are four main classes of hypervolume-based AAs based on the guarantees concerning the selected μ solutions with respect to the parent population of μ solutions

[2]It is important to note that a tighter approximation bound was derived [103] very recently. The new bound relies both in n and k while the previous $(1-1/e)$ bound did not. A weak but simple form of the new bound is: $1-\left(1-\frac{m}{k}\right)\left(1-\frac{1}{k}\right)^{k-m}$. It is tighter for $k > \frac{n}{2}$ by acknowledging that in that case a greedy and an optimal solution must agree in m points where m is at least $2k-n$. Therefore, any reference to the old $(1-1/e)$ bound is also valid for the new bound.

and the current set of λ offspring solutions:

1. **Nondecreasing**: The hypervolume of the μ selected solutions is equal or higher than the hypervolume of the parent population.

2. **Increasing**: The hypervolume of the μ selected solutions is strictly higher than the hypervolume of the parent population or equal if there is no subset of μ solutions with higher hypervolume.

3. **Locally Optimal**: The hypervolume of the μ selected solutions is the maximum with respect to all μ-sized subsets among all solutions from the parent population and the offspring.

4. **(Decremental) Greedy**: The μ solutions are selected using a decremental greedy algorithm, i.e., by iteratively discarding the least contributor until only μ solutions remain.

Note that Bringmann and Friedrich [36] refer to archiving algorithms using a decremental greedy algorithm just as "greedy" archiving algorithm. Therefore, their results do not directly apply to any other type of greedy algorithm such as incremental ones (see Section 3.4.5 on "Greedy Algorithms").

From the definitions, it is clear that any Locally Optimal AA is also Increasing and any Increasing AA is also Nondecreasing, although the reverse is not true. A $(\mu + \lambda)$-archiving algorithm that uses an algorithm to solve HSSP exactly (see Section 3.4.4), where $k = \mu$ and $n = \mu + \lambda$, is a Locally Optimal AA and consequently, an Increasing and a Nondecreasing AA. The decremental greedy $(\mu + \lambda)$-AA is also Locally Optimal if $\lambda = 1$ (i.e., $n = \mu + 1$ and $k = \mu$). On the contrary, if $\lambda > 1$ an AA purely based on a decremental greedy algorithm is not even guaranteed to be Nondecreasing. However, if the subset selected by the decremental greedy algorithm is only accepted by the archiver if it does not have lower hypervolume than the parent population, then the archiver is Nondecreasing. Such greedy archivers will be referred to as Nondecreasing Decremental Greedy $(\mu + \lambda)$-AA.

3.3.2 Convergence Analysis

The convergence of Archiving Algorithms can be analyzed based on effectiveness and on competitiveness. The former focuses on how close an archiving algorithm can get to the (hypervolume of an) optimal subset of the Pareto front maximizing the HSSP for $k = \mu$ (optimistic analysis). The latter, also referred to as a measure of regret, focuses on how far an archiving algorithm can get from the μ-sized subset of the set of all visited points with higher hypervolume (pessimistic analysis). Both are independent of the underlying methods used for creating the initial population and for generating offspring. The successive offspring generations are seen as a sequence of arbitrary λ-sized solution sets. In both analysis, a worst-case initial population is considered, i.e., any μ-sized set of feasible solutions can be the initial population. A best case would be the optimal solution which is not considered of interest. For the sequence of offspring sets, a best case is considered for the effectiveness analysis, while a worst case is considered for the competitiveness analysis. Formally:

Archiving Algorithm	**Convergence behavior**
Nondecreasing $(\mu+\lambda)$, $\lambda<\mu$	There is no $\left(1+0.1338\left(\frac{1}{\lambda}-\frac{1}{\mu}\right)-\epsilon\right)$-approximate, $\epsilon>0$
Increasing $(\mu+\lambda)$, $\lambda=\mu$	Effective
Locally optimal $(\mu+\lambda)$, $\lambda=\mu$	Effective
Increasing $(\mu+\lambda)$, $\lambda<\mu$	$(2-\lambda/\mu+\epsilon)$-approximate, $\epsilon>0$
Locally optimal $(\mu+\lambda)$, $\lambda<\mu$	$(2-\lambda/\mu+\epsilon)$-approximate, $\epsilon>0$
Nondecreasing $(\mu+\lambda)$	There is no $(1.1338-0.1338/\mu-\epsilon)$-competitive, $\epsilon>0$
Increasing $(\mu+\lambda)$	There is no $(\mu-\epsilon)$-competitive, $\epsilon>0$
Increasing $(\mu+\lambda)$, $\mu=1$	μ-competitive
Locally optimal $(\mu+\lambda)$	μ-competitive

Table 3.1: Convergence of $(\mu+\lambda)$-archiving algorithms

Effectiveness An archiving algorithm is said to be effective iff, for any initial population, there is always a sequence of offspring sets such that an optimum is achieved, i.e., a population corresponding to a subset that maximizes the hypervolume indicator for μ solutions is always found.

α-approximate If an archiving algorithm is not effective, one can ask how close to the optimum it can get. In such cases, an archiving algorithm is said to be α-approximate iff, for any initial population, there is always a sequence of offspring sets such that a population with at least $1/\alpha$ times the hypervolume of an optimal μ-sized subset is reached.

α-competitive An archiving algorithm is said to be α-competitive, iff for any initial population and any sequence of offspring sets (that may explore just a small portion of the objective space), a population with, at least, $1/\alpha$ times the hypervolume of the optimal μ-sized subset among all visited points is always reached. This gives an indication whether the archiver is able to consistently maintain the best subset of points among all points seen.

Ideally, an archiving algorithm would be effective. That would mean that, whichever the initial population is, there is always a sequence of offspring such that the AA is able to find an optimal μ-sized subset. Moreover, it is desirable that an algorithm is α-competitive such that α is (close to) one, which would mean that the AA never loses the best subset among all visited points. However, intuitively, such property does not seem achievable for $\alpha=1$ in general, as the choice of solutions in the optimal subset for HSSP depends on the available solutions to choose from, i.e., without any knowledge of future solutions.

Theoretical Convergence Summary

Table 3.1 summarizes the theoretical convergence results stated by Bringmann and Friedrich [36]. Note that all results for Increasing AAs are also valid for the corresponding Locally Optimal AAs.

Looking at Table 3.1 it is possible to state that a Locally Optimal (and increasing) $(\mu+1)$-AA, such as SMS-EMOA [16], although not effective (unless $\mu=1$), it is $(2-\frac{1}{\mu}+\epsilon)$-approximate, for any $\epsilon>0$. This means that, for every problem, there is a sequence of offspring for which the ratio between the hypervolume of SMS-EMOA last population and the hypervolume of the optimal μ-sized subset is better than $(2-\frac{1}{\mu}+\epsilon)^{-1}$. However, there is no nondecreasing $(\mu+\lambda)$-AA for $\lambda < \mu$ such that it is β-approximate, where $\beta = \left(1 + 0.1338\left(\frac{1}{\lambda} - \frac{1}{\mu}\right) - \epsilon\right)$. This means that there is, at least, a problem instance and an initial population for which $\frac{1}{\beta}$ is an upper bound for the approximation ratio of the hypervolume of SMS-EMOA final population. That is, given that λ is 1, then the ratio is lower than $\frac{1}{\beta} < \frac{1}{1.1338} \simeq 0.88199$ if μ is large and lower than $\frac{1}{\beta} < \frac{1}{1.0669} \simeq 0.93729$ if $\mu = 2$.

Although results on effectiveness for nondecreasing $(\mu+\lambda)$-AAs with $\lambda \geq \mu$ are unknown, those AAs that are also Increasing are effective. Moreover, unless $\mu = 1$, there is no Increasing, and consequently no Locally optimal $(\mu+\lambda)$-AA that is 1-competitive, i.e., there is always an initial population and an offspring sequence such that the AA does not maintain the best μ-sized subset and thus, does not reach an optimal solution. Using $\lambda \geq \mu$ for locally optimal AAs is suggested in [36] to ensure that the archiver has the ability to reach an optimal population.

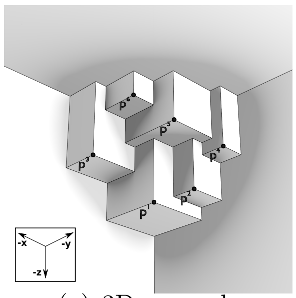
Regarding Decremental Greedy $(\mu+\lambda)$-AA, the convergence guarantees for increasing and locally optimal AAs hold if $\lambda = 1$, otherwise no guarantees apply unless the Nondecreasing version is considered.

3.4 Algorithms

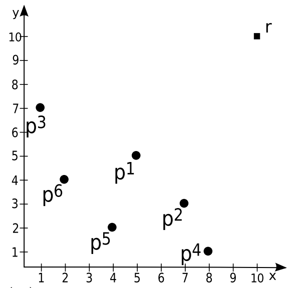
This section gives an overview of the state-of-the-art algorithms for the hypervolume-related problems described in Section 3.1. The techniques and paradigms used in such algorithms are described first (see Subsection 3.4.1). The following four subsections provide an overview of the existing algorithms for computing the hypervolume indicator, hypervolume contributions, the HSSP and greedy approximations to the HSSP, respectively. More focus is given to techniques and algorithms related to those that will be proposed in Section 4 through detailed and illustrative examples. These illustrative examples are also intended to introduce the use of 2-dimensional projections of 3-dimensional example problems. This type of illustration is extensively used in Section 4 to help explain the proposed algorithms.

In order to make the referencing of algorithms easier, the state-of-art algorithms are identified by a name and also alphanumerically. A letter and a number (e.g.[A2]) is assigned to each algorithm. The letter identifies to which problem the algorithm relates to (A – hypervolume indicator, B – hypervolume contributions, C – HSSP, D – greedy approximation to the HSSP) and the number is assigned according to the order in which they are introduced in the following sections. For example, algorithm [B4] is the fourth algorithm related to hypervolume contributions introduced (Section 3.4.3). Links to available implementations are given with the description of each algorithm.

For the following chapter, a complete understanding of the techniques described in Section 3.4.1 is not required except for the Dimension-Sweep paradigm. The same is true for the algorithms, although understanding HV3D, HV4D (see [A4] and [A6] in Section 3.4.2, respectively), EF and UHV3D (see [B2] and [B5] in Section 3.4.3, respectively) is recommended. For a quick overview of the available and the fastest algorithms both with respect to runtime and to asymptotic complexity, see the “Remarks” at the end of Sections 3.4.2 to 3.4.5.



(a) 3D example



(b) 2D projection at $z = r_z$

p	point
p^1	$(5, 5, 1)$
p^2	$(7, 3, 2)$
p^3	$(1, 7, 4)$
p^4	$(8, 1, 5)$
p^5	$(4, 2, 6)$
p^6	$(2, 4, 8)$

Figure 3.5: Three-dimensional base example and the corresponding projection on the (x, y)-plane. The reference point is $r = (10, 10, 10)$.

3.4.1 Techniques

Figure 3.5 shows a 3-dimensional example that is used for illustration purposes throughout this section. Moreover, n is used to represent the input size.

Inclusion-Exclusion Principle

The inclusion-exclusion principle is a technique consisting of sequentially iterating over an inclusion step followed by an exclusion step, where the i^{th} step involves some computation for every combination of i points. This technique is discussed in [8] for the Hypervolume: The hypervolume indicator of a set $\mathrm{S} \subset \mathbb{R}^d$ of n points is the sum of the hypervolume (indicator) of every subset with a single point, minus the hypervolume of the component-wise maximum of each pair of points, plus the hypervolume of the component-wise maximum of each subset with three points, minus the hypervolume of the component-wise maximum of each subset of four points, and so on. This technique can be very inefficient, if applied as explained above since it is exponential in the number of points, $\Theta(2^n)$.

Dimension Sweep

Dimension sweep is a paradigm [113] which has been widely used in the development of algorithms for hypervolume-related problems (e.g. [A2], [A8], [A3], [A4], [A6]). A problem involving n points in $\mathbb{R}^d$ is solved with this paradigm by visiting all points in ascending (or descending) order of one of the coordinates, solving a $(d-1)$-dimensional subproblem for each point visited, and combining the solutions of those subproblems. The subproblems themselves can often be solved using dimension sweep as well, until a sufficiently low-dimensional base case is reached, which can be solved easily by a dedicated algorithm. However, the time complexity of the resulting algorithms typically increases by an $O(n)$ factor per dimension.

A typical dimension-sweep algorithm for Hypervolume problem works as follows. Input points are sorted and visited in ascending order of the last coordinate. The d-dimensional dominated region is partitioned into n slices by axis-parallel cut hyperplanes defined by the last coordinate value of each input point and the reference point. The desired hypervolume indicator value is the sum of the hypervolumes of all slices, and the hypervolume of a slice is the hypervolume of its $(d-1)$-dimensional base multiplied by its height. The base of a slice is the $(d-1)$-dimensional region dominated by the projection of the points below it according to dimension d onto the corresponding cut hyperplane. The height of a slice is the difference between the values of the last coordinate of two consecutive points.

Figure 3.6(a) exemplifies how the volume in the example of Figure 3.5 could be split in $n = 6$ (horizontal) slices. This division and the computation of the volume for this example is further detailed in the explanation of HV3D (see [A4]) in Section 3.4.2. Figure 3.6(b) shows a splitting of the base area of the topmost slice in Figure 3.6(a).

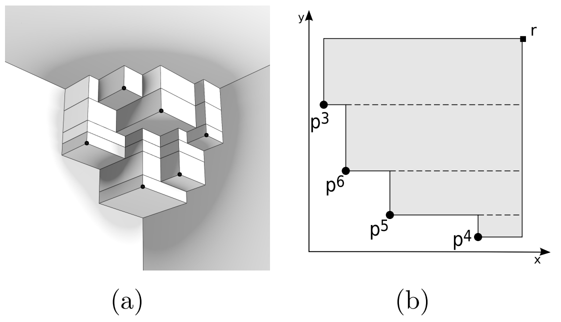


Figure 3.6: Example of: (a) the slice division of the volume in Figure 3.5(a) and; (b) of the area of the corresponding topmost slice.

The algorithms for hypervolume-based problems using dimension-sweep differ mostly in the $(d-1)$-dimensional subproblem considered. For example, for the Hypervolume problem, one option would be to compute the $(d-1)$-dimensional hypervolume indicator of the base of the slice from scratch or avoid the full computation by updating the hypervolume of the base of the previous slice.

Spatial Divide-and-Conquer

This technique consists of splitting the d-dimensional hypervolume into two parts and recursively solving each part until a problem easy to solve is reached. The problem is split into two subproblems according to the median point, p, of a given dimension i, i.e., the $\lceil (n+1)/2 \rceil^{th}$ point with lowest coordinate i. The axis parallel hyperplane at the value p_i of coordinate i divides the hypervolume in two parts. The first part refers to a subproblem containing the $\lfloor n/2 \rfloor$ points below p in the i^{th} coordinate and the reference point is the one of the current problem but projected on the splitting hyperplane. The second subproblem contains all n points, but the $\lfloor n/2 \rfloor$ points below p in the i^{th} coordinate are projected onto the splitting hyperplane. Figure 3.7 shows an example of these two subproblems where coordinate $i = 3$ is used for splitting and the median point p^4 is the splitting point p.

This approach is used, for example, by HOY algorithm (see [A7] in Section 3.4.2) to compute Hypervolume and by Bringmann and Friendrich's algorithm (see [B1] in Section 3.4.3) to compute AllContributions. An orthogonal partition tree is typically used as the underlying data structure, as in the mentioned algorithms. In that case, the hyperrectangle bounded below by the component-wise minimum of the initial point set and above by the reference point is recursively partitioned in axis parallel regions, and each one of them is associated to a node. Moreover, the typical base case of the recursion (a leaf of the partition tree) is when a partition

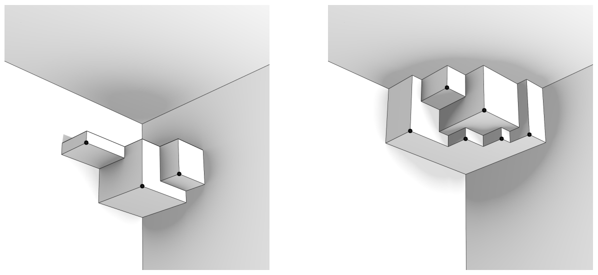

(a) First subproblem (b) Second subproblem

Figure 3.7: Example of the volume division in Figure 3.5(a) using the spatial divide-and-conquer approach.

consisting of a trellis is reached. A trellis is a region (hyperrectangle) where every point in the subproblem dominates that region in all coordinates except one [14].

Multidimensional Divide-and-Conquer

The Multidimensional Divide-and-Conquer paradigm [13] is a well-known paradigm in Computational Geometry and is used for problems such as computing the maxima (or minima) of a point set and finding the closest pair [13]. In such cases, a d-dimensional problem with n points is divided in three subproblems, two d-dimensional subproblems of size $n/2$ and a $(d-1)$-dimensional subproblem of size n (the merge step). Each of the subproblems is solved recursively using the same approach until a problem easy to solve is reached.

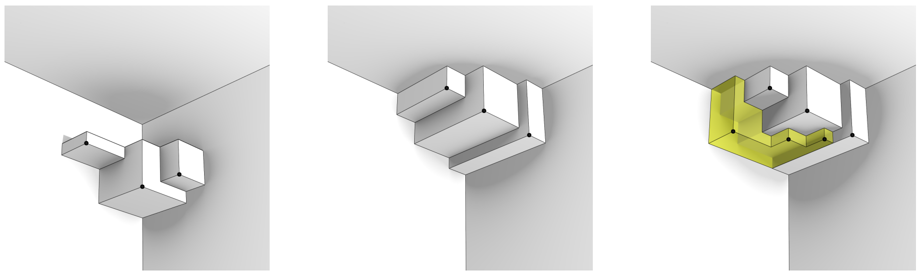
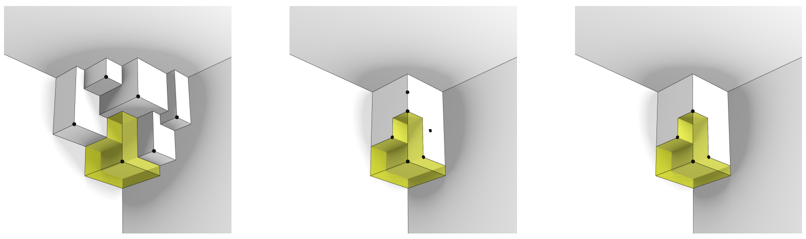

(a) First subproblem (b) Second subproblem (c) Merge step

Figure 3.8: Example of the volume division in Figure 3.5(a) using the multidimensional divide-and-conquer approach.

Figure 3.8 gives an example of the three possible subproblems for the hypervolume indicator. Similarly to the Spatial Divide-and-Conquer, the median point of a given coordinate i can be used for splitting. However, in this case, the second subproblem only considers the $\lceil n/2 \rceil$ points with equal or higher coordinate i than the splitting point. The third subproblem consists of computing the contribution of the set of points in the first subproblem with respect to the set of points in the second subproblem. HVDC3D [78] (see [A5] in Section 3.4.2) is the only algorithm for hypervolume-based problems that uses this paradigm.

Bounding Technique

The bounding technique is the technique of projecting points onto the surface of an axis parallel d-dimensional box [26, 34] and discarding the points that become dominated, in d-dimensional space. In practice, it consists of determining the auxiliary set J in the definition of delimiter (see Definition 3.5), where J is the smallest set of points weakly dominated by p that delimits its contribution. It is used in several algorithms (e.g. [129, 128, 105]), in particular, to compute hypervolume contributions.



(a) Hypervolume Contribution (b) Bounding (c) Filtering

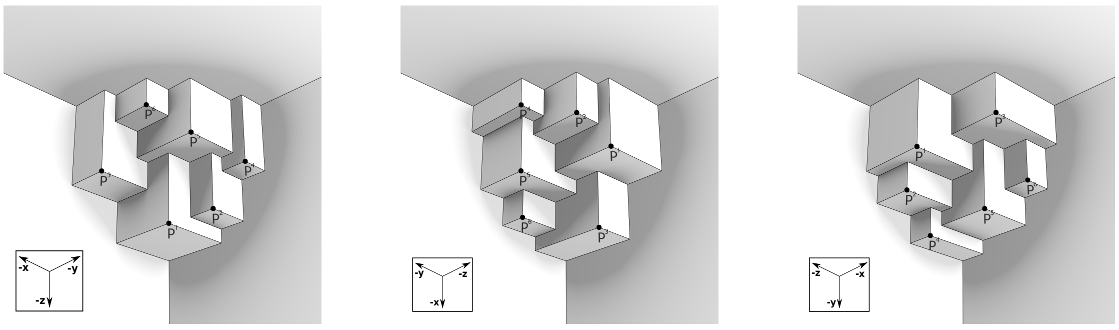
Figure 3.9: Example of the bounding technique for the contribution of p^1, in Figure 3.5(a).

See Figure 3.9 for an example. Figure 3.9(a) shows the hypervolume of a set of points $\text{S} = \{p^2, \ldots, p^6\}$ from Figure 3.5 and the hypervolume contribution of p^1 (in transparent yellow). Figure 3.9(b) shows the bounding technique, where all points (not dominated by p^1) are projected on the surface of the region dominated by p^1, and where the gray axis-parallel box shown is the volume dominated by such projections. Then, only the nondominated points among those projected points are kept (see Figure 3.9(c)).

This technique is used in the computation of hypervolume contributions because these projections are enough to delimit the contribution of p^1 and because the absolute position of the delimiters resulting in those projections is irrelevant, i.e., have no influence on the contribution of p^1. Moreover, this bounding technique allows to further discard the points that are not delimiters of the contribution of p^1 as they are unnecessary to compute it.

Objective Reordering

Problems in computational geometry such as those related to the hypervolume indicator are invariant with respect to objective reordering and, in particular, to the case where it is equivalent to rotation. See the example in Figure 3.10 of a volume in (x, y, z) coordinate system (see Figure 3.10(a)) and its counter-clockwise (see Figure 3.10(b)) and clockwise rotations (see Figure 3.10(c)), corresponding to the (y, z, x) and (z, x, y) coordinate systems, respectively. Reordering objectives is important since, although the result of the computation does not change, it can have an impact on the implementation runtime [130]. For example, Dimension-Sweep approaches are usually sensitive to objective ordering as a particular order may result in more dominated points further in the recursion than others. While *et al.* [130] proposed several heuristics to determine the best objective order to consider. Algorithms such as WFG (see Section 3.4.2) benefit with the integration of such heuristics, by repeating it in several steps of the computation of the hypervolume indicator, which results in fast algorithms in practice in many data set instances.



(a) (x, y, z) (b) (y, z, x) (c) (z, x, y)

Figure 3.10: Example of Figure 3.5(a) considering three of the six objective order possible.

Dynamic Programming

Dynamic Programming (DP) is used for solving combinatorial optimization problems that can be broken down into subproblems which overlap and which have optimal substructure. The optimal substructure property guarantees that the optimal solution to a problem can be constructed from the optimal solution to its subproblems [48]. A DP approach consists of splitting a problem into several subproblems, solving each of them if it has not been solved before and store its solution in order to efficiently look it up the next time the solution to that subproblem is required. Each of the subproblems is usually further divided in a similar way, until a small problem easy to solve, a base case, is reached.

Dynamic Programming is used in algorithms to solve the HSSP. In an HSSP problem considering an initial set of n points and k as the subset size desired, the solution obtained with this technique is, for example, constructed based on the solution of $O(n)$ subproblems of size $k-1$. Similarly, the solution of each subproblem of size $k-1$ may be constructed based on the solutions to $O(n)$ subproblems of size $k-2$ and so on. See Section 3.4.4 for more algorithmic details.

Local Upper Bounds

The orthant containing a given set of n nondominated points in $\mathbb{R}^d$ and limited above by the reference point $r \in \mathbb{R}^d$ consists of two disjoint regions, the region dominated by such a set and the *search region* [95]. In multiobjective optimization, the latter is the promising region where to look for (more) optimal solutions. Recall that the hypervolume indicator can be defined as the union of boxes bounded below by a point in the nondominated point set and above by the reference point. Analogously, the search space is also defined by the union of boxes which are unbounded below and bounded above by *local upper bounds* [93]. The set of local upper bounds can be roughly defined as the nondominated point set of the search region considering maximization (see Figure 3.11). These local upper bounds can be computed from the given set of n points and the reference point. In $d = 2$ there are $n + 1$ local upper bounds and in $d = 3$ there are $2n + 1$ [52]. In the general $d \geq 2$ case there are $\Omega(n^{d/2} \log n)$ upper bounds [93, 95]. Such local upper bounds can be determined and used for the computation of the hypervolume indicator (see [A11] in Section 3.4.2).

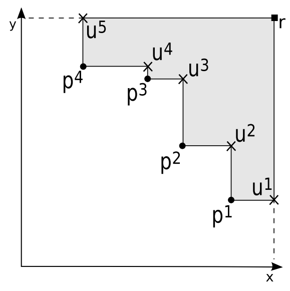


Figure 3.11: Example of the points which are the local upper bounds (points $u^1, \ldots, u^5$ represented by crosses, $\times$) given the nondominated point set $\{p^1, \ldots, p^4\}$.

Branch-and-Bound

In a typical branch-and-bound approach to a combinatorial maximization problem, the algorithm recursively solves a given problem by dividing it (branching) in one or more subproblems and avoiding the computation of subproblems that definitely will not lead to optimal solutions (through bounding). Every recursion call fixes one more (set of) component(s) of the solution and thus represents the subproblem associated with a partial solution. To avoid further recursions, an upper bound on the best solution value achievable from such a partial solution is computed. If such a bound is worse (is lower) than the best solution computed so far, than that subproblem is skipped. Such approach is used to solve HSSP and/or HSSPComplement (see [C7] in Section 3.4.4).

3.4.2 Hypervolume Indicator

There are many algorithms to compute the Hypervolume problem. This section overviews only the most conceptually distinct ones (that are based on a different paradigm/technique) and the fastest ones according either to their asymptotic complexity or to their runtime efficiency. The list of algorithms excluded from this overview include some algorithms based on problems for which the Hypervolume is a particular case (e.g. [29, 46, 134]) and others specific to the Hypervolume computation (e.g. [8, 27]), approximation algorithms (e.g., HypE [10]) and algorithms based on parallelization (e.g., [105, 119]). Of all algorithms presented, the descriptions of HV3D and HV4D are the most detailed due to their importance for the next chapter.

Algorithms

[A1] Lebesgue Measure Algorithm (LebMeasure)
The LebMeasure algorithm [62] was one of the first algorithms proposed to compute Hypervolume for any number of dimensions ($d > 1$). The algorithm maintains a list of points, $S \subset \mathbb{R}^d$, initially set to the input point set. The first step is to remove a point p from S and to compute and accumulate the hypervolume of an

hypercube inside the region exclusively dominated by p. This is the hypercube bounded below by p and bounded above in every coordinate $i \in \{1, \ldots, d\}$ by b_i, the lowest i-th coordinate value in the point set $\{q \in \mathrm{S} \mid p_i \leq q_i\}$. To account for the part of the contribution of p to S not yet computed, d new points are added to S, each one is a projection of p onto an axis-parellel hyperplane at b_i in coordinate i. Then, dominated points are removed from S. These steps are repeated until S is empty. Although LebMeasure was presented as a polynomial-time algorithm, its time complexity was later shown to be exponential in d: $O(n^d)$ [127].

[A2] Hypervolume by Slicing Objectives (HSO)
The HSO algorithm [97, 131][3] is a direct application of the dimension sweep approach to the computation of Hypervolume problem. HSO works exactly as explained in the previous section (see under "Dimension sweep") and has $O(n^{d-1})$-time complexity. Several algorithms were proposed based on HSO, considering different $(d-1)$-dimensional subproblems, data structures and/or base cases or by combining it with other techniques (e.g., FPL [A3] and WFG [A8]).

[A3] Fonseca, Paquete and López-Ibañez's algorithm (FPL) [4]
The FPL algorithm [68] is based on HSO and has $O(n^{d-2} \log n)$ time complexity and $O(n)$ space complexity. FPL improves upon HSO through the use of more efficient data structures, the caching of previous computations and the use of a better base case for the recursion. In particular, FPL maintains points sorted according to every dimension in circular doubly linked lists. In an FPL recursion call, points are removed from (some) lists in decreasing order of a given dimension and are reinserted in reverse order. This behavior enables constant time insertions. Moreover, when a new point is visited in an i-th dimensional subproblem, it adds contribution only to the $(i-1)$-th dimensional slices above it in dimension $i-1$. Thus, if $(i-1)$-th dimensional slices below that point have been previously computed then FPL has that information stored and does not need to recompute them as HSO does. Finally, FPL stops the recursion at $d = 3$ where it uses HV3D (see [A4]).

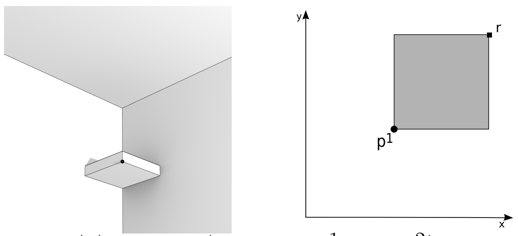
[A4] Beume *et al.*'s algorithm for $d = 3$ (HV3D) [5]
HV3D [15] is a dimension-sweep algorithm for the $d = 3$ case with $\Theta(n \log n)$ time complexity and $O(n)$ space complexity. Given a point set $\mathrm{S} = \{p^1, \ldots, p^n\} \subset \mathbb{R}^3$, the Hypervolume problem is computed in HV3D by dividing the volume in slices (see the example in Figure 3.6(a)) from bottom up as detailed in Figure 3.12. Each pair of figures show the three-dimensional representation of a slice and its two-dimensional base (on the (x, y)-plane).

Assume that $p^1_z \leq p^2_z \leq \ldots \leq p^n_z$. Every point marks the beginning of the slice immediately above it and the closing of the slice immediately below it (with the exception of the lowest point in z-coordinate). Consequently, the height of the i-th slice is $p^{i+1}_z - p^i_z$, where $i = 1, \ldots, n-1$, and the height of the last slice is $r_z - p^n_z$ (see Figure 3.12). The base of a slice is delimited only by the set of points below it that are nondominated on the (x, y)-plane. In the example (see Figure 3.12(e)), p^1 and p^2 do not delimit the base of slice 5 because p^5 delimits it and dominates both of them

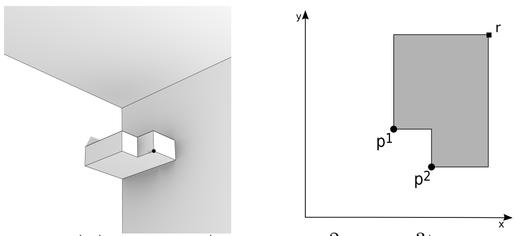
[3]The algorithm was proposed in [97] and was later named and studied in more detail in [131]. While *et al.* [131] also assign independent authorship of such an algorithm to E. Zitzler, and provide a reference to source code (`ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c`).

[4]An implementation is made available by the authors: `http://lopez-ibanez.eu/hypervolume`

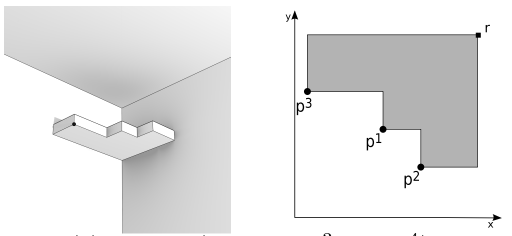
[5]An implementation is made available by the authors with FPL [A3]



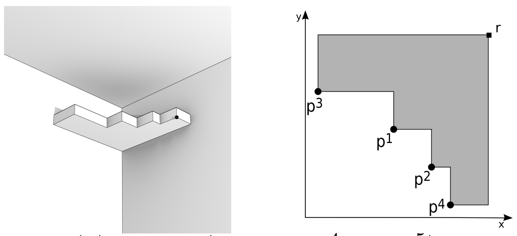
(a) Slice 1 (between p_z^1 and p_z^2)



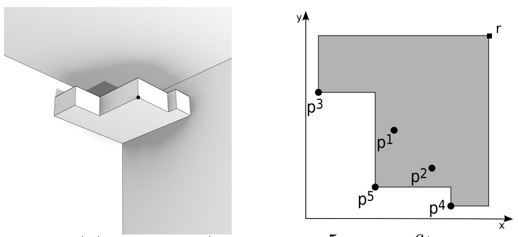
(b) Slice 2 (between p_z^2 and p_z^3)



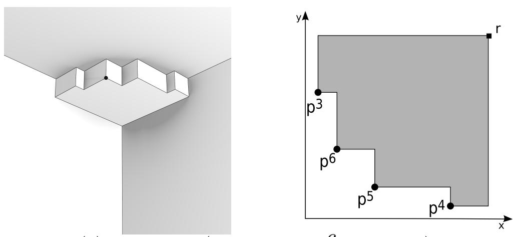
(c) Slice 3 (between p_z^3 and p_z^4)



(d) Slice 4 (between p_z^4 and p_z^5)



(e) Slice 5 (between p_z^5 and p_z^6)



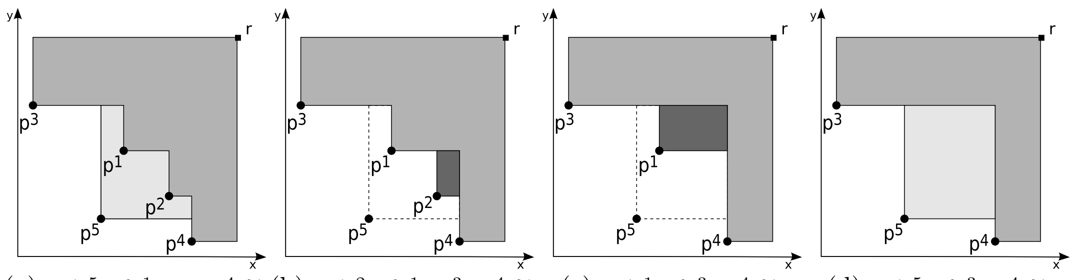
(f) Slice 6 (between p_z^6 and r_z)

Figure 3.12: The splitting of the volume in Figure 3.5(a) into 6 slices and the corresponding 2-dimensional bases.

in their 2-dimensional projection. Therefore, p^4, p^5 and p^6 are enough to compute the base of slice 5 and p^1 and p^2 can be discarded as they do not delimit slice 5 nor any slice above it. The hypervolume indicator of $\mathrm{S} = \{p^1, \ldots, p^6\}$ is therefore:

$$\begin{aligned} H(\mathrm{S}) = {} & (p_z^2 - p_z^1)\ H(\{p^{1*}\}) \\ & + (p_z^3 - p_z^2)\ H(\{p^{1*}, p^{2*}\}) \\ & + (p_z^4 - p_z^3)\ H(\{p^{1*}, \cdots, p^{3*}\}) \\ & + (p_z^5 - p_z^4)\ H(\{p^{1*}, \cdots, p^{4*}\}) \\ & + (p_z^6 - p_z^5)\ H(\{p^{3*}, \cdots, p^{5*}\}) \\ & + (r_z - p_z^6)\ H(\{p^{3*}, \ldots, p^{6*}\}) \end{aligned}$$

Instead of explicitly computing the base area of each slice from scratch as expressed above, HV3D computes them more efficiently by computing two-dimensional contributions. Every point $p \in \mathrm{S}$ dominates the area $[(p_x, p_y), (r_x, r_y)]$ between p_z and r_z, and so it dominates that area in all slices above p according to z-coordinate. Therefore, the base area of the slice that begins at $z = p_z$ can be computed as the base area of the slice that ends at $z = p_z$ plus the contribution of p^* to the points delimiting the base of that slice (see Figure 3.12(a) to 3.12(f)). Therefore, $H(\mathrm{S})$ can



(a) $H(p^{5*}, \{p^{1*}, \ldots, p^{4*}\})$ (b) $H(p^{2*}, \{p^{1*}, p^{3*}, p^{4*}\})$ (c) $H(p^{1*}, \{p^{3*}, p^{4*}\})$ (d) $H(p^{5*}, \{p^{3*}, p^{4*}\})$

Figure 3.13: Update of the base area of slice 5 (see Figure 3.12(e)).

be computed as:

$$\begin{aligned} H(\mathrm{S}) = &(p_z^2 - p_z^1)\ H(p^{1*}, \{\}) \\ &+ (p_z^3 - p_z^2)\ (H(\{p^1\}) + H(p^{2*}, \{p^{1*}\})) \\ &+ (p_z^4 - p_z^3)\ (H(\{p^{1*}, p^{2*}\}) + H(p^{3*}, \{p^{1*}, p^{2*}\}) \\ &+ (p_z^5 - p_z^4)\ (H(\{p^{1*}, \cdots, p^{3*}\}) + H(p^4, \{p^{1*}, \cdots, p^{3*}\})) \\ &+ (p_z^6 - p_z^5)\ (H(\{p^{1*}, \cdots, p^{4*}\}) + H(p^5, \{p^{1*}, \cdots, p^{4*}\})) \\ &+ (r_z - p_z^6)\ (H(\{p^{3*}, \cdots, p^{5*}\}) + H(p^6, \{p^{3*}, \cdots, p^{5*}\})) \end{aligned}$$

Note that the left-hand side of the sum term in each line is the result of the sum on the right-hand side of the multiplication in the previous line (e.g., $H(\{p^{1*}, \cdots, p^{3*}\}$ in the fourth line is equal to the sum $H(\{p^{1*}, p^{2*}\}) + H(p^{3*}, \{p^{1*}, p^{2*}\})$ in the previous line). HV3D solves the Hypervolume problem in this way, by computing the slices from bottom up and, by storing and updating the base of the last slice computed. This corresponds to solving a sequence of 2-dimensional UpdateHypervolume problems.

In detail, HV3D works as follows. Given an n-point set $\mathrm{S} \subset \mathbb{R}^3$, points in S are sorted and visited in ascending z-coordinate order. Each point $p \in \mathrm{S}$ marks the beginning of a new slice, the base area of which is computed by updating the area of the base of the previous slice (if it exists). This is illustrated in Figure 3.13(a), where the darker gray region represents the base of the previous slice to be updated when $p = p^5$ is visited (slice 4 depicted in Figure 3.12(d)). To that end, the points visited so far whose projections on the (x, y)-plane are mutually nondominated are kept sorted in ascending order of the y coordinate using a height-balanced binary tree, T. In the example of Figure 3.13(a), $\mathrm{T} = \{p^4, p^2, p^1, p^3\}$. For each $p \in \mathrm{S}$, the point $q \in \mathrm{T}$ with $q_y < p_y$ and the least q_x such that $q_x > p_x$ (p^4 in the example) is determined in $O(\log n)$ steps by searching T. Then, the contribution of p^* to T^* is computed by visiting the successors of q in T in ascending order of y until a point is found whose projection is not dominated by p^* (in the example, p^3).

Let the set of points in T that are weakly dominated by p on the (x, y)-plane be denoted by $\mathrm{Q} = \{q \in \mathrm{T} \mid p^* \leq q^*\}$. Points in Q are visited in ascending order of y-coordinate and for each point $q \in \mathrm{Q}$, the contribution of q^* to T^* is computed and subtracted from the area of the base of the previous slice in constant time, and q is removed from T in $O(\log n)$ time. In the example of Figure 3.13(a), $\mathrm{Q} = \{p^2, p^1\}$ and therefore, the contribution of $p^* = p^{5*}$ to $\mathrm{T}^* = \{p^{1*}, \ldots, p^{4*}\}$ is computed as:

$$H(p^{5*}, \mathrm{T}^*) = -H(p^{2*}, \{p^{1*}, p^{3*}, p^{4*}\}) - H(p^{1*}, \{p^{3*}, p^{4*}\}) + H(p^{5*}, \{p^{3*}, p^{4*}\})$$

The contribution of p^{2*} to $\mathrm{T}^* = \{p^{1*}, \ldots, p^{4*}\}$ (see the darker region in Figure 3.13(b)) is computed and p^2 is removed from T. Then, the contribution of p^{1*} to $\mathrm{T}^* = \{p^{1*}, p^{3*}, p^{4*}\}$ (see the darker region in Figure 3.13(c)) is computed and p^1 is removed from T. Afterwards, the area of the region exclusively dominated by p^*, $(p^4_x - p_x) \times (p^3_y - p_y)$ (see Figure 3.13(d)), is determined and added to the current base area. Finally, p is added to T in $O(\log n)$ time once the base area has been updated. The volume of the slice is computed and T contains the points delimiting its base, i.e., $\mathrm{T} = \{p^4, p^5, p^3\}$.

In the above, each point in S is visited twice: once when it is added to T and again when it is removed from T. Since all of the corresponding operations are performed in $O(\log n)$ time, the algorithm has amortized $O(n \log n)$ time complexity.

[A5] Divide-and-Conquer algorithm for $d = 3$ (HVDC3D) [6]
HVDC3D [78], as HV3D, has optimal $\Theta(n \log n)$ time complexity and $O(n)$-space complexity for the $d = 3$ case of Hypervolume problem. It directly applies the Multidimensional Divide and Conquer paradigm as explained in the previous section.

[A6] Guerreiro *et al.*'s algorithm for $d = 4$ (HV4D) [7]
HV4D [84, 78] is an $O(n^2)$-time and $O(n)$-space algorithm for the particular case of $d = 4$ of the Hypervolume problem. Although it has worse time complexity than Chan's algorithm [A9], it is currently the fastest one among the algorithms with available implementations.

HV4D algorithm is an extension of HV3D to four dimensions where a sequence of three-dimensional UpdateHypervolume problems is solved via the corresponding OneContribution problems using similar techniques to those in the EF algorithm (see [B2]). Points in the input set $\mathrm{S} \subset \mathbb{R}^4$ are visited in ascending order of the last coordinate, partitioning the dominated region into four-dimensional slices. For each $p \in \mathrm{S}$, the base volume of the new slice is computed by updating the volume of the base of the previous slice with the contribution of p^* to the projection on (x, y, z)-space of the points visited so far.

For that purpose, the points visited so far whose projections are nondominated are stored in a data structure, L. The contribution of each $p^* \in \mathrm{S}^*$ to L^* is computed in linear time using UHV3D (see [B5]), provided that points in L are sorted in two lists in ascending order of the y and z coordinates, respectively and, provided that $\mathrm{L}^* \cup \{p^*\}$ is a nondominated point set. However, even though S is a nondominated point set, S^* may not be and consequently, p^* may dominate some points in L^*. To fulfill the last requirement of UHV3D, the contribution $H(p^*, \mathrm{L}^*)$ is computed in a similar way as in HV3D. Hence, one by one, each point $q \in \mathrm{L}$ such that $p^* \leq q^*$ is removed from L and the contribution of q^* to L^* is computed using UHV3D, which is then subtracted from the volume of the previous slice. Afterwards, the contribution $H(p^*, \mathrm{L}^*)$ is computed and added to the current volume and p is added to L.

Since a three-dimensional contribution is computed at most twice for each input point, once when it is added to L and once in case it is removed from L, then $O(n)$ calls to UHV3D are performed and consequently, the time complexity of HV4D amortizes to $O(n^2)$.

[6] There is an implementation of this algorithm but is available only upon request to the author.

[7] An implementation is made available by the authors: `https://github.com/apguerreiro/HV4D`.

[A7] Hypervolume Overmars and Yap (HOY) [8]
HOY algorithm [14] has $O(n^{d/2} \log n)$ time complexity, although it was initially thought to have $O(n^{d/2})$ time complexity [17] due to a gap in the analysis and which is acknowledged in [14]. It is based on an algorithm for Klee's measure problem by Overmars and Yap and uses a streaming variant of an orthogonal partition tree as the underlying data structure. Only $O(n)$ space complexity is required by constructing the referred tree on-the-fly instead of storing it completely as in the classical variant which would require $O(n^{d/2})$ space complexity.

HOY is based on the Spatial Divide-and-Conquer paradigm (see the previous section). It measures the hypervolume dominated by a given point set $\mathrm{A} \subset \mathbb{R}^d$ inside the region bounded by a lower and an upper reference point, ℓ and u, respectively. The set A is initialized as the set of input points, u to the given reference point and ℓ to the component-wise minimum of all input points. HOY recursively partitions the considered space, firstly using dimension one. In a given call, let i denote the last dimension used to partition the current partition. Then, HOY checks if dimension i satisfies certain criteria for partitioning, otherwise, it checks dimension $i+1$ and so on. The recursion base case is achieved when the (d-dimensional) dominated region of the current partition forms a trellis. Moreover, before HOY is used, points are previously sorted according to dimension d. This allows to shrink the space partition considered in each recursive call by visiting points in A in ascending order of dimension d until a point q that dominates the $(d-1)$ projection of the space partition is found. In such case, the hypervolume of the space region between q_d and u_d is computed right away and u_d can be set to q_d.

[A8] Walking Fish Group algorithm (WFG) [9]
WFG algorithm [129] is currently one of the fastest for computing the hypervolume indicator in many dimensions, particularly for $d > 7$ (see the experimental results in [102]), even though it is not asymptotically the fastest. WFG was initially reported to have $O(2^{n+1})$ time complexity [129] but Renaud *et al.* [102] recently tightened this upper bound to $O(n^{d-1})$ and presented a lower bound of $\Omega(n^{d/2} \log n)$.

WFG is an algorithm mainly based on the bounding technique and on ideas from the inclusion-exclusion principle, and is further optimized by integrating ideas of dimension-sweep and objective reordering. In particular, WFG works as follows. Points in a given set $\mathrm{X} \subset \mathbb{R}^d$ are sorted and visited in ascending order of dimension d. For each point $p \in \mathrm{X}$ visited, the $(d-1)$-dimensional contribution of p to the set of already visited points, S (i.e., $H(p^*, \mathrm{S}^*)$), is then computed and multiplied by the difference $(r_d - p_d)$. The hypervolume of S is the sum of all such multiplications. The contribution $H(p^*, \mathrm{S}^*)$ is computed by subtracting to the hypervolume of $\{p^*\}$ the hypervolume of the set S^* but bounded by p^*, i.e., $H(p^*, \mathrm{S}^*) = H(p^*, \mathrm{J}) = H(\{p^*\}) - H(\mathrm{J})$, where J is the set obtained with the bounding technique (see Section 3.4.1). In its turn, $H(\mathrm{J})$ is computed recursively with WFG. In practice, WFG alternates between OneContribution and Hypervolume problems.

The bounding technique in combination with dimension sweep allows many points to become dominated and thus plays an important role in reducing the required computational effort. This helps placing WFG among the fastest for many

[8]An implementation is made available by the authors: `ls11-www.cs.tu-dortmund.de/people/beume/publications/hoy.cpp`

[9]An implementation is made available by the authors: `http://www.wfg.csse.uwa.edu.au/hypervolume/`

dimensions ($d > 4$).

[A9] Chan's algorithm (Chan)
Chan's algorithm [47] has $O(n^{d/3} \text{ polylog } n)$ time complexity which is currently the best time complexity to compute Hypervolume for $d \geq 4$. Chan's paper proposes an algorithm for the general case of Klee's Measure problem and then derives an algorithm for the special case of the hypervolume indicator (referred as the union of arbitrary orthants). Chan's algorithm is a spatial divide-and-conquer algorithm combined with a procedure to simplify partitions that is repeated every few levels of cutting, reducing the number of boxes a partition contains. One of the main differences to HOY is the method used for selecting the dimension to be used for partitioning which changes at every level of recursion, first it uses dimension one, then dimension two, and so on.

Despite its good time complexity, no implementation was found available online.

[A10] Quick Hypervolume (QHV) [10]
QHV [118] is an algorithm for the general case of Hypervolume problem that is based on quicksort. This algorithm can be viewed as a type of spatial divide-and-conquer but instead of dividing the hypervolume in two as discussed in the example given in Section 3.4.1, QHV splits it in 2^d regions. To compute Hypervolume for a point set $\mathrm{S} \subset \mathbb{R}^d$, first a point $p \in \mathrm{S}$ is selected to take the role of a pivot. The point in S with greatest contribution to the empty set is selected for this purpose. The pivot splits the d-dimensional space in $O(2^d)$ hyperoctants sharing a corner at p and p is discarded. The two hyperoctants referring to the regions that contain the points dominating p or the points that p dominates are discarded. QHV is called recursively for each of the remaining hyperoctants (still partially dominated by S) with the points inside that region and the projection on the hyperplanes delimiting the hyperoctant of the points that partially dominate it. QHV includes a subrotine to discard dominated points from each hyperoctant. The hypervolume of S is the sum of the hypervolume returned from the recursive calls plus the contribution of p to the empty set. In base cases with up to 10 points, a simple algorithm as HSO or based in the Inclusion-Exclusion principle is used.

In the worst-case, QHV has $O(n(d + \log^{n-2})2^{nd})$ time complexity and $O(n^d n^2)$-space complexity. However, its performance depends on the characteristics of the data set considered. For example, the authors showed that the time complexity on data sets where points are uniformly distributed over a hypersphere is $O(dn^{1.1} \log^{n-2} n)$. In practice, QHV was observed to be competitive with WFG. Although a parallel version of QHV exists [119], only the sequential version is taken into account in this thesis.

[A11] Hypervolume Box Decomposition Algorithm (HBDA) [11]
Lacour *et al.* [102] proposed an algorithm to compute Hypervolume in any number of dimensions. This algorithm, HBDA, computes the hypervolume indicator by partitioning the dominated region into $O(n^{\lfloor \frac{d}{2} \rfloor})$ axis-parallel boxes and adding up the corresponding hypervolumes. The partitioning results from computing all local upper bounds, where each box is associated to one local upper bound. The incremental version of the algorithm (HBDA-I) runs in $O(n^{\lfloor \frac{d}{2} \rfloor + 1})$ time and is char-

[10] An implementation is made available by the authors: `http://web.tecnico.ulisboa.pt/luis.russo/QHV/#down`

[11] An implementation is made available by the authors: `https://github.com/renaudlr/hbda`

acterized by computing a sequence of n UpdateHypervolume problems, allowing input points to be processed in any order. Since the current box decomposition must be stored across iterations, $O(n^{\lfloor \frac{d}{2} \rfloor})$ space is required. By processing input points in ascending order of any given coordinate, the memory requirements are reduced to $O(n^{\lfloor \frac{d-1}{2} \rfloor})$, and the time complexity is improved to $O(n^{\lfloor \frac{d-1}{2} \rfloor + 1})$. HBDA-NI (the non-incremental version) has been shown to be competitive in $d \geq 4$ dimensions, but its memory requirements are a limiting factor for large d.

[A12] Cox and While algorithm for $d = 4$ (HV4DX) [12]
HV4DX was proposed by Cox and While [50] as an alternative version of HV4D to compute Hypervolume for $d = 4$. In HV4DX, for each point $p \in \mathrm{S} \subset \mathbb{R}^4$ visited, the contribution of p^* to the set of previously visited points, L^*, is also partitioned in axis-parallel boxes. The differences lie on how this partitioning is done and in not having the requirement that $\mathrm{L}^* \cup \{p^*\}$ is a nondominated point set. Without such requirement, HV4DX does not have to previously remove the points dominated by p^* nor to recompute the corresponding contributions, as HV4D does. Although HV4DX is claimed to improve upon HV4D, the experimental results presented in Section 4.1.3 do not support those claims.

Remarks

Table 3.2 summarizes the described algorithms. It indicates for how many dimensions they can be used for (d) and for which they are recommended (d') based on their runtime performance and/or time complexity. Moreover, it indicates whether an implementation of such algorithms is available online and which are the paradigms/techniques used (see Section 3.4.1) by each one of them. The bottom part of Table 3.2 summarizes the best algorithms regarding time complexity and/or runtime performance (based on the experimental results presented in [118, 102]).

As an optimal $\Theta(n \log n)$-time algorithm and the fastest in practice for $d = 3$, HV3D is the recommended algorithm for $d = 3$. For $d \geq 4$, Chan's algorithms is recommended as it has the best time complexity, but, because no implementation is available and there is no information on how it performs in practice, other algorithms are recommended as alternatives. For $d = 4$, HV4D is the fastest one. For $d = 5, 6$, HBDA-NI [102] held the best runtimes while for $d \geq 7$ it is not always clear which is the fastest one among WFG, QHV and HBDA-NI, particularly because their ranking seems to be strongly dependent of the input data. Since HBDA-NI only outperformed the other two in the experiments in [102] in a specific data set that is particularly difficult for WFG (see the "Hard" Data set in Section 3.5) and because its memory requirements grow exponentially with d, HBDA-NI is not recommended for $d \geq 7$. In such cases, WFG and QHV are preferable. It is unclear which of the two is the fastest, because the experimental results presented in [118] concluded that QHV is the fastest while in [102] WFG was the fastest one.

Note that, most of the currently fastest algorithms available (HV3D, HV4D and WFG) are all dimension-sweep based. Each one solves a sequence of subproblems smaller in the number of dimensions and in a way that avoids recomputing everything from scratch and/or that tries to reduce the problem size. For example, HV3D and HV4D solve Hypervolume by iterating over UpdateHypervolume

[12]An implementation is made available by the authors: `http://www.wfg.csse.uwa.edu.au/hypervolume/#code`

Algorithm	d	d'	Time-complexity	Available	Characteristics
LebMeasure [A1]	≥ 2	-	$O(n^d)$	?	-
HSO [A2]	≥ 2	-	$O(n^{d-1})$	Yes	DS
HVDC3D [A5]	3	-	$O(n \log n)$	No*	MDC, DS
HOY [A7]	≥ 2	-	$O(n^{d/2} \log n)$	Yes	SDC, DS
Chan [A9]	≥ 4	≥ 4	$O(n^{d/3}\text{polylog } n)$	No	SDC
FPL [A3]	≥ 2	2	$O(n^{d-2} \log n)$	Yes	DS
HV3D [A4]	3	3	$\Theta(n \log n)$	Yes	DS
HV4D [A6]	4	4	$O(n^2)$	Yes	DS
HBDA-NI [A11]	≥ 2	$5, 6$	$O(n^{\lfloor \frac{d}{2} \rfloor})$	Yes	LUBs
WFG [A8]	≥ 2	≥ 7	$\Omega(n^{d/2} \log n)$	Yes	IE, B, DS
QHV [A10]	≥ 2	≥ 7	$O(n(d + \log^{n-2})2^{nd})$	Yes	SDC

Table 3.2: Algorithms for the Hypervolume problem. (DS - Dimension-sweep, IE - Inclusion-Exclusion, B - Bounding Technique, SDC - Spatial Divide-and-Conquer, MDC - Multidimensional Divide-and-conquer, LUBs - Local Upper Bounds).

(incremental scenario) while WFG alternates between OneContribution and Hypervolume problems and takes advantage of the bounding technique, and by doing so they all reduced the computational costs required when compared to HSO.

3.4.3 Hypervolume Contributions

Hypervolume-based selection is typically related to the AllContributions and/or the HSSP problems, while the Hypervolume problem is more related to the performance evaluation of EMOAs. The computation of hypervolume contributions is frequently required either directly by the EMOA selection method (e.g., in the special case of $k = n - 1$ of HSSP, or for ranking), or indirectly, in the inner steps of hypervolume-related algorithms (e.g., in algorithms to approximate the HSSP). Although algorithms for Hypervolume can also be used to compute contributions by solving a sequence of Hypervolume problems, it is typically more advantageous to use algorithms particular to such problems. Thus, the algorithms in Section 3.4.2 should be used only as last resort, when there is no problem-specific alternative.

This section focuses on algorithms for problems related to the exact computation of hypervolume contributions, in particular: AllContributions, OneContribution, LeastContributor and the UpdateAllContributions problem. Only the state-of-the-art algorithms are described. Those (non-competitive) algorithms purely based on HSO (e.g., [136]) and approximation algorithms (e.g., [10, 33]) were left out. Due to their importance to this thesis, EF and UHV3D are explained in more detail.

Algorithms for AllContributions problem

[B1] Bringmann and Friedrich algorithm (BF1)

Bringmann and Friedrich [35] proposed an algorithm for the HSSPComplement problem (see BF [C1]). Given a point set $X \subset \mathbb{R}^d$ of n points and a subset size k, the algorithm computes the contribution to X of every subset of $n - k$ points. In the particular case of $k = n - 1$, the algorithm computes the contribution of every subset of size 1, i.e., the AllContributions problem. For this particular case, the algorithm has $O(n^{d/2} \log n)$ time complexity and will be here referred to as BF1.

[B2] Emmerich and Fonseca algorithm (EF) [13]
EF is a dimension-sweep algorithm proposed by Emmerich and Fonseca [61] for the $d = 3$ case of the AllContributions problem and has $\Theta(n \log n)$ time complexity and $O(n)$ space complexity. This algorithm extends HV3D to the computation of the contributions of all points in a nondominated set $\mathrm{X} \subset \mathbb{R}^3$. As in HV3D, EF visits all points in X in ascending order of z coordinate while maintaining a balanced binary tree T. This tree plays the same role as in HV3D, that is, to maintain the visited points (nondominated on the (x, y)-plane) that delimit the base of the current slice. These are now also the points whose contributions are being computed, and when the projection of a point becomes dominated and the point is removed from T, it also means that its contribution is fully computed. Therefore, in addition to the data structure used in HV3D, for each point $p \in \mathrm{X}$ visited, a box partition of the contribution of p^* to T^* is stored in a doubly-linked list, and the box partitions of the corresponding delimiters in T^* are updated. Individual boxes are characterized by their lower and upper corners in all three dimensions, although they are initially unbounded above in the z dimension.

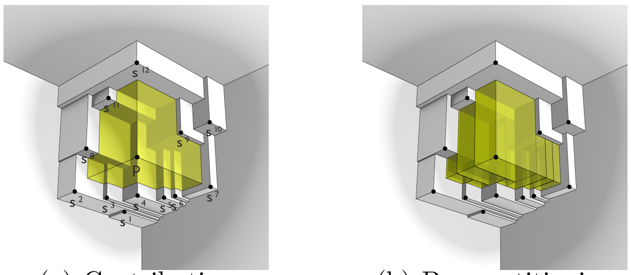
Boxes are stored until another point in X that (partially) dominates them on the (x, y)-plane is visited. In particular, for each $p \in \mathrm{X}$, all boxes associated with the inner delimiters of p^* are closed. Closing a box means setting the z coordinate of its top corner to p_z, computing its volume and adding it to the current value of the contribution of the associated point, and discarding that box. Boxes partially dominated by p^* are also closed, and are replaced by a new box accounting for the base area that remains not dominated by p^*. All box operations are performed in $O(1)$ time, and since at most $O(n)$ boxes are created and closed, the algorithm retains the $O(n \log n)$ time complexity of HV3D. UHV3D also does the partitioning of the contribution of a point in boxes in the same way as EF. See in the description of UHV3D ([B5]) a detailed and illustrated explanation of this box partitioning.

Emmerich and Fonseca [61] also proposed a $\Theta(n \log n)$ algorithm for the $d = 2$ case. In this case, it is easy to see that all (box-shaped) contributions can be computed in linear time after sorting X.

[B3] Exclusive (Contribution) Quick Hypervolume (exQHV) [14]
The authors of QHV (see [A10]) extended the algorithm for the AllContributions problem in any number of dimensions [119]. The main differences to QHV are: 1) the need to store all contributions instead of a single value; 2) the point used as pivot and; 3) the exclusion of dominated points. In the case of QHV, the point p with greatest contribution to the empty set is the pivot. Because the contribution of p also has to be computed, the authors decided not to use p as pivot, but to use instead the point p' with greatest contribution to the empty set among the points obtained from the coordinate-wise maximum between p and each point in the hyperoctant. In QHV, all dominated points in an hyperoctant are discarded. However, in exQHV, points dominated by a single point are needed for computing the contribution of the point dominating it and therefore, only the points dominated by two

[13] An implementation is made available by the authors: `http://liacs.leidenuniv.nl/~csnaco/index.php?page=code`

[14] An implementation is made available by the authors: `http://web.tecnico.ulisboa.pt/luis.russo/QHV/#down`



(a) Contribution (b) Box partitioning

Figure 3.14: Example of a contribution, $H(p, \mathrm{S})$, and of its division in boxes, where $\mathrm{S} = \{s^1, \ldots, s^{12}\}$.

or more points are excluded from each hyperoctant. As in QHV, an algorithm such as HSO or based on the Inclusion-Exclusion principle is used for small cases with up to 10 points. Such algorithms were also adapted to compute contributions.

Algorithms for OneContribution problem

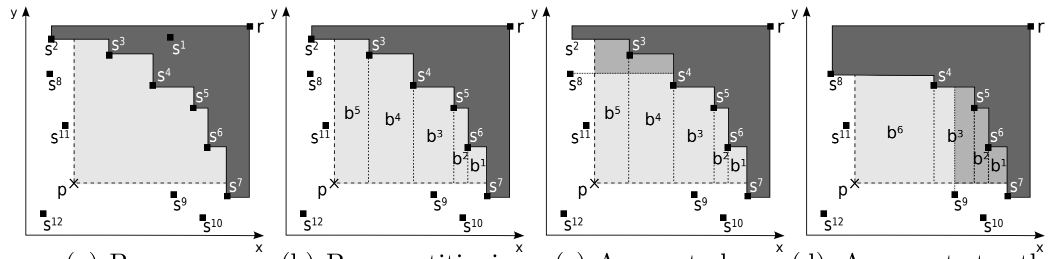
[B4] Incremental HSO (IHSO)
IHSO [24, 25] is an algorithm for computing the contribution of a single point, $p \in \mathbb{R}^d$, to a set of points $\mathrm{X} \subset \mathbb{R}^d$, i.e., for the OneContribution problem. It is based on HSO, but only the contribution of p to X is split into slices whose hypervolumes are summed up at the end. Points that are not delimiters of the contribution of p to X are ignored. Additionally, an objective reordering heuristic is used to choose a good order in which to process the objectives. IHSO works even if some points in X are dominated by p.

There is an improved version of IHSO that uses FPL's data structure (see [A3]). It was proposed as part of IIHSO [27] which is an algorithm for the Hypervolume problem that uses IHSO to iteratively solve OneContribution problems.

[B5] Guerreiro *et al.*'s update algorithm (UHV3D) [15]
HV4D algorithm [A6] iterates over a procedure to solve the incremental case of UpdateHypervolume for $d = 3$ by solving a OneContribution problem in $d = 3$. This procedure is interesting by itself and relevant for this thesis, which justifies its detachment from HV4D. The standalone version of this procedure will be called here as UHV3D. This approach computes the contribution of a point $p \in \mathbb{R}^3$ to a point set $\mathrm{S} \subset \mathbb{R}^3$ though its correctness and efficiency is guaranteed only if: 1) $\mathrm{S} \cup \{p\}$ is a nondominated point set and; 2) S is previously stored in a data structure consisting of two linked lists sorted according to y- and z-coordinates, respectively.

UHV3D computes a three-dimensional contribution in the same way as EF does for all contributions, i.e., by partitioning it into axis-parallel boxes and by summing up their volumes. Figure 3.14(a) shows an example of the contribution of p to $\mathrm{S} = \{s^1, \ldots, s^{12}\}$ and Figure 3.14(b) shows how it is partitioned in boxes (8, in this case). The projections on the (x, y)-plane in Figure 3.15 illustrate the first steps of the computation of such contribution. The contribution is computed in two steps. The two-dimensional base of the contribution (see Figure 3.15(a)) is first partitioned into boxes, by sweeping the points in S in ascending order of y. Then, all points

[15]An implementation made available by the authors integrated in HV4D [A6] code



(a) Base area (b) Base partitioning (c) Area cut above (d) Area cut to the right

Figure 3.15: Example of the box update in the computation of the contribution from the example in Figure 3.14, projected on the (x, y)-plane. Note that $s^1_z < \ldots < s^7_z < p_z < s^8_z < \ldots < s^{12}_z$.

$q \in \mathrm{S}$ such that $q_z > p_z$ are visited in ascending order of z, and for each of these points, the boxes that are (partially) dominated by q^* are updated.

First, consider that S is split into two sets, $\mathrm{S}^1 = \{q \in \mathrm{S} \mid q_z \leq p_z\}$ and $\mathrm{S}^2 = \{q \in \mathrm{S} \mid q_z > p_z\}$. Then, the delimiters of the contribution of p to S^1 on the (x, y)-plane are stored in S' by sweeping S^1 in ascending order of the y-coordinate. In the example, $\mathrm{S}^2 = \{s^8, \ldots, s^{12}\}$, $\mathrm{S}^1 = \{s^1, \ldots, s^7\}$ and $\mathrm{S}' = \{s^2, \ldots, s^7\}$. By visiting the points in S' in ascending order of y-coordinate, a sorted list of $|\mathrm{S}'| - 1$ non-overlapping boxes is constructed. The z-coordinate of the bottom of every box is set to p_z and the top is left unbounded. Figure 3.15(b) shows the boxes associated to point p at $z = p_z$. Then, points $q \in \mathrm{S}^2$ are visited in ascending order of the z-coordinate and for each one, the list of boxes is updated so as to represent the contribution of p^* at $z = q_z$. To that end, every box b (partially) dominated by q on the (x, y)-plane is closed, i.e., its top is set to q_z, its lower left corner, b^l, is set to $(\max(b^l_x, q_x), \max(b^l_y, q_y), b^l_z)$ and the box is discarded after its volume is computed and added to the contribution computed so far. Afterwards, a new (single) box is added to account for the region not dominated by q^* and which was covered by the recently discarded boxes before the lower corner update. For example, in Figure 3.15(c), $q^* = s^{8*}$ partially dominates boxes b^4 and b^5 and so box b^6 is added (see Figure 3.15(d)). When $q = s^9$ (see Figure 3.15(d)), boxes b^1,b^2, b^3 are closed and a single box is added to account the area of b^3 not dominated by s^{9*}. When all boxes are closed (in the example, all boxes are closed after processing $q = s^{12}$), the contribution of p to S is fully computed.

A box is only inserted at, and removed from, the beginning or the end of the list of boxes. Therefore, all operations on boxes are performed in constant time and a maximum of $O(n)$ boxes is created and closed. Splitting S takes $O(n)$, however, it is used here only for clarity. Visiting points in S in ascending order of y (and then z) and skipping those that do not fulfill the conditions of S' (of S^2) also takes $O(n)$ time. Hence, the contribution of p to S is computed in amortized $O(n)$ time.

As a standalone algorithm to compute the OneContribution problem for $d = 3$, UHV3D is suitable for the incremental scenario of UpdateHypervolume and for the decremental scenario as well. Moreover, UHV3D can also be used for the UpdateAllContributions problem, although n calls would be required.

Algorithms for LeastContributor

[B6] Incremental HSO* least contributor algorithm (IHSO*)
IHSO* [24] is an algorithm based on IHSO (see [B4]) for the LeastContributor problem. In the search for the least contributor, IHSO* tries not to have to fully compute all contributions. Of all tested schemes, Bradstreet *et al.* [24] concluded that the best to avoid full computations is to use an objective reordering heuristic combined with Best-First-Queueing (BFQ). At each step of BFQ, the point with the currently lowest (partially) computed contribution is picked and a “bit” more of its contribution is computed. A parameter of BFQ defines how much a “bit” more is. This process continues until a point with its contribution fully computed is picked and this is the least contributor. However, the notion of a “bit of contribution” has to be reasonably defined, i.e., if the considered granularity is too large, then all contributions are computed, if too small, then an excessive number of iterations is required. The time complexity of IHSO* is not reported. IHSO* authors also proposed an algorithm to update the least contributor under single-point changes [22], i.e., to identify the (new) least contributor after adding or removing a point.

[B7] Incremental WFG least contributor algorithm (IWFG) [16]
IWFG [128, 51] is an algorithm for the computation of the LeastContributor and is the result of a combination between WFG (see [A8]) and IHSO* (see [B6]). It uses the best-first queuing mechanism of IHSO* to gradually update contributions until the least contributor is found. Therefore, in IWFG, the contribution of each point is divided in slices and the first slice associated to each point is computed using WFG. Then, repeatedly, the point with the lowest (partially) computed contribution is picked and its next slice is computed also using WFG. The algorithm stops when the selected point is one whose contribution is fully computed (is the least contributor). Cox and While [51] proposed an improved version of IWFG that uses a new slicing scheme and reordering of objectives for each point which led to better runtimes and to outperform IHSO*. The time complexity of IWFG is not reported.

Algorithms for UpdateAllContributions

[B8] Hupkens and Emmerich update algorithm for 2D (UHVC2D)
Hupkens and Emmerich [89] proposed an algorithm, here called UHVC2D, to update the contributions of every point in a set $X \subset \mathbb{R}^2$ to the set X itself under single-point changes to X (incremental and decremental cases of UpdateAllContributions), in the $d = 2$ case. This algorithm has $O(n)$ space complexity and $O(\log n)$ time complexity provided that X is a nondominated set and is previously stored sorted along a coordinate in a balanced binary tree. Igel *et al.* [90] had previously proposed an update method very similar to UHVC2D. However, it was proposed as a part of a procedure for ranking solutions and only the decremental scenario was contemplated.

Note that there are at most two delimiters of the contribution of a point $p \in X$ to a nondominated set $X \subset \mathbb{R}^2$ and each point in X only delimits the contribution of its delimiters. Therefore, UHVC2D does a $O(\log n)$-time search in the tree storing X to find the delimiters of the contribution of the point p being added/removed. The contribution of each delimiter and of p is then updated in $O(1)$ time. The incremental

[16] An implementation made available by the authors: `http://www.wfg.csse.uwa.edu.au/hypervolume/`

case has an extra step to remove points in X dominated by the new point before the contribution update. As the tree insertion/deletions are performed in $O(\log n)$, UHVC2D performs in amortized $O(\log n)$ time per point. Moreover, note that the algorithm could be easily applied to the UpdateHypervolume problem for $d = 2$.

Remarks

Table 3.3 summarizes the information regarding the algorithms described in this section. It reports the best algorithms for each problem and the number of dimensions for which they can be used, the corresponding time complexities (if known) and indicates whether an implementation is available online. In the latter case, a "Yes*" means that an implementation is not exactly available but is easy to implement/obtain. In the case of UHVC2D, the algorithm is easily implemented. In the case of UHV3D and WFG, although the algorithms were not originally proposed for solving the indicated problem, with some (simple) modifications/adaptations to HV4D and WFG, respectively, they can be used for that purpose. In the case of WFG, recall that it explicitly solves several OneContribution problems in its inner steps (see [A8]). The code for WFG available online can be easily modified to make use of that inner step to compute just the OneContribution and even to compute the AllContributions problem. This is more advantageous than iterating over the original WFG and use it to compute every contribution as the difference between two hypervolumes (see Definition 3.3). Finally, the reported time complexities of UHVC2D and UHV3D assume that the data structures were previously set up. In the incremental scenario, these complexities represent the worst-case complexity if $\{p\} \cup \mathrm{S}$ is a nondominated point set. Otherwise, if p may dominate points in S, then they represent the amortized complexities per point for a sequence of n calls (i.e., n incremental updates to an initially empty set S).

For the AllContributions problem, EF and BF1 are the recommended algorithms for $d = 3$ and $d > 3$, respectively. As there is no available implementation of BF1, exQHV and WFG are both recommended as alternatives. For the OneContribution problem, UHV3D and WFG are the recommended algorithms for $d = 3$ and $d > 3$, respectively. For the LeastContributor problem and $d \geq 2$, IWFG is recommended, however, it may be more advantageous to use an algorithm for the AllContributions problem and then identify the least contributor (for example, in the experiments in Section 4.2.3 EF was more efficient than IWFG for $d = 3$). For the UpdateAllContributions problem, UHVC2D is recommended for $d = 2$, while for $d > 2$ the best alternative is to use the algorithms recommended for the AllContributions problem. Due to the absence of algorithms dedicated to the explicit computation of the AllContributions2 and UpdateAllContributions2 problems, the recommendation is to iterate over algorithms for OneContribution. In the case of $d = 3$, computing either problem would have a cost of $O(mn)$ after sorting R, where $n = |\mathrm{R}|$ and $m = |\mathrm{S}|$ (see Definitions 3.4 and 3.8) using UHV3D.

On dominated points Although most algorithms presented so far are expected to be used only for nondominated point sets, they may have to somehow deal with dominated points. For example, HV3D (see [A4]) and UHV3D (see [B5]) have to compute OneContribution problems in $d = 2$ (for the base of the contribution

Problem	Algorithm	d	Time-complexity	Available
AllContributions	BF1 [B1]	≥ 2	$O(n^{d/2} \log n)$	No
	EF [B2]	$2, 3$	$O(n \log n)$	Yes
	exQHV [B3]	≥ 2	-	Yes
	WFG [A8]	≥ 2	$\Omega(n^{d/2} \log n)$	Yes*
OneContribution	IHSO [B4]	≥ 2	-	No
	UHV3D [B5]	3	$O(n)$	Yes*
	WFG [A8]	≥ 2	$\Omega(n^{d/2} \log n)$	Yes*
LeastContributor	IHSO* [B6]	≥ 2	-	No
	IWFG [B7]	≥ 2	-	Yes
UpdateAllContributions	UHVC2D [B8]	2	$O(\log n)$	Yes*

Table 3.3: Algorithms for computing hypervolume contributions related problems.

of the new point) which consists of computing $H(p, \mathrm{S})$ where $p \in \mathbb{R}^2$ and $\mathrm{S} \subset \mathbb{R}^2$ and where p may dominate some points in S (see the example if Figure 3.12). Many algorithms use a work-around. In HV3D (and also in HV4D, but for $d = 3$), one by one, each point in S dominated by p (the set of all is $\mathrm{Q} = \{q \in \mathrm{S} \,|\, p \leq q\}$) is removed from S in $O(\log n)$, its contribution to S is computed and afterwards $H(p, \mathrm{S} \setminus \mathrm{Q})$ is computed. Even though the time complexity of OneContribution computation amortizes to $O(\log n)$ per input point along HV3D, a single OneContribution computation has $O(n \log n)$ worst-case time complexity (similarly, the worst case time complexity of a single call to UHV3D is $O(n^2)$ if dominated points are allowed). Moreover, the information concerning the inner delimiters of the contribution of p to S is lost. In algorithms where a contribution must be successively updated, this information is required, and thus dominated points have to be dealt differently. For example, in order to update a 2-dimensional contribution, UHV3D keeps a list of boxes that implicitly store information concerning all of its delimiters, allowing it to have full information concerning the exact shape of the contribution of p.

3.4.4 Exact Algorithms for the HSSP

Although the computation of the hypervolume indicator raised a lot of interest in the last decade and better algorithms have successively been proposed (see Section 3.4.2), there have been fewer contributions for the HSSP, which is a **NP**-hard problem. There is currently no algorithm for $d > 3$ that does not check every combination of k points. Only for $d = 2, 3$, are there a few algorithms that avoid that much computation. The existing algorithms to compute the HSSP exactly are described next.

For the algorithm description, let $\mathrm{X} \subset \mathbb{R}^d$ be a nondominated point set such that $|\mathrm{X}| = n$ and that an optimal subset of size $k \in \{1, \dots, n\}$ out of those n points needs to be selected. Let $r \in \mathbb{R}^d$ be a reference point such that every point in X strongly dominates it.

Algorithms

The first algorithm for the general d-dimensional case for the exact computation of the HSSP was proposed in 2010 [35] and explicitly checks every combination of $(n - k)$ points to discard. Most of the remaining algorithms are focused on the particular cases of 2 and 3 dimensions. In 2009, a $O(kn^2)$-time dynamic programming algorithm was proposed for the $d = 2$ case [3]. Faster algorithms were proposed only recently, one in 2014 by Bringmann and Friedrich [37] with $O(nk + n \log n)$ time complexity, and two other in 2016 by Kuhn *et al.* [100] of which the asymptotically fastest one has $O(k(n-k) + n \log n)$ time complexity. The algorithms by Kuhn *et al.* [100] rely on alternative formulations of the HSSP, one as a k-link shortest path (k-LSP) problem and another as an Integer Linear Programming (ILP) problem. These are interesting formulations even though the former works only for the 2-dimensional case and the generalization of the latter to any number of dimensions [99, 76] has a number of constraints exponential in d.

More recently, Bringmann *et al.* [31] proposed an algorithm for $d = 3$ with $O(n^{\sqrt{k}})$ time complexity, while for the general d-dimensional case, Gomes [75, 76] proposed a branch-and-bound algorithm. Apart from these two very recent algorithms (from 2017) that will be listed and only briefly characterized at the end, all other algorithms referred above are described next in some detail, first the general algorithm and then those specific to the low-dimensional cases ($d = 2, 3$).

[C1] General Algorithm by Bringmann and Friedrich (BF)
The algorithm by Bringmann *et al.* [35] solves the HSSPComplement problem, i.e., it finds the subset $\mathrm{S} \subseteq \mathrm{X}$ of $(n-k)$ points that contributes the least to the hypervolume of the whole set X. This algorithm will be called BF here. As HOY (see [A7]), BF is an adaptation of Overmars and and Yap's algorithm. It uses a partition tree to store the leaf partitioning (of a spatial divide-and-conquer). The contribution to X of every subset of $(n-k)$ points are explicitly computed by summing up, among every leaf, the hypervolume indicator of the regions associated to each of these subsets. The algorithm has a $O(n^{d/2} \log n + n^{n-k})$ time complexity and $O(\min(n^{d/2}, n^{n-k}))$-space complexity [31]. Consequently, this algorithm is adequate for high values of k (small $n-k$), in particular for $n - k \leq d/2$, but for small values of k it may become too expensive. For $n - k = 1$, the algorithm solves the AllContributions problem (see BF1 [B1]).

[C2] $O(kn^2)$ **Dynamic Programming Algorithm (DPHSS)**
Auger *et al.* [3] proposed a bottom-up Dynamic Programming (DP) algorithm for the 2-dimensional SSP problem with respect to the weighted hypervolume indicator, of which HSSP is a special case. The algorithm explained next focuses only on this particular case and will be referred to here as DPHSS. DPHSS has $O(kn^2)$ time complexity[17] and $O(kn)$-space complexity. However, if only the maximal hypervolume value of a k-sized subset is needed and not the subset itself then only $O(n)$-space complexity is required.

Given a nondominated point set $\mathrm{X} = \{p^1, \ldots, p^n\} \subset \mathbb{R}^2$ where $p^j_x < p^{j+1}_x$ for $j = 1, \ldots, n-1$ without loss of generality, and given a subset size $k \in \{1, \ldots, n\}$, let (i, t) denote the subproblem of finding a subset $\mathrm{S} \subset \mathrm{X}$ maximizing the hypervolume indicator such that $\mathrm{S} \subseteq \{p^i, \ldots, p^n\}$, $p^i \in \mathrm{S}$ and $|\mathrm{S}| \leq t$, where $i \in \{1, \ldots, n\}$ and $t \in \{1, \ldots, k\}$. Let $P(i, t)$ denote an optimal solution to subproblem (i, t) and let subproblems $(i, 1)$ and (n, t) be the base cases with optimal solutions $P(i, 1) = \{p^i\}$ and $P(n, t) = \{p^n\}$, respectively. For $t > 1$ and $i < n$, the optimal substructure property lies in the fact that if the leftmost point is removed from an optimal solution

[17]The time complexity reported in the original paper is $O(n^3)$ [3] but $O(kn^2)$ is a tighter upper bound, as reported in [37].

$\mathrm{S} = P(i,t)$ to the subproblem (i,t), then the obtained subset $\mathrm{S} \setminus \{p^i\}$ is an optimal solution to subproblem $(j, t-1)$ where p^j denote the leftmost point in $\mathrm{S} \setminus \{p^i\}$. Thus, an optimal solution to subproblem (i,t) is obtained from $P(i+1, t-1), \ldots, P(n, t-1)$ by finding, in linear time, a $j \in \{i+1, \ldots, n\}$ for which $H(\{p^i\} \cup P(j, t-1))$ is maximized. Note that such a hypervolume is computed in constant time. Computing the solution to subproblems $(1,t), \ldots, (n,t)$ thus takes $O(n^2)$ time for each $t \in \{1, \ldots, k\}$ leading, together with the initial sorting, to an overall $O(kn^2 + n \log n)$ time complexity. The optimal solution to the HSSP is given by the best solution among $P(1,k), \ldots, P(n,k)$.

[C3] HypSSP [18]
Bringmann *et al.* [37] proposed HypSSP, an $O(n(k + \log n))$-time algorithm for the 2-dimensional HSSP. Analogously to DPHSS, HypSSP requires $O(n)$-space complexity if only the hypervolume value of the optimal subset(s) is required, and $O(nk)$-space complexity if such subset is also needed. The description of HypSSP in [37] considers maximization, but to preserve the consistency and clarity, the description here considers minimization.

Given a nondominated point set $\mathrm{X} = \{p^1, \ldots, p^n\} \subset \mathbb{R}^2$ where $p^j_x < p^{j+1}_x$ for $j = 1, \ldots, n-1$ without loss of generality, the reference point $p^{n+1} = r$ and given a subset size $k \in \{1, \ldots, n\}$, let (i,t) denote the subproblem of finding a subset $\mathrm{S} \subseteq \{p^1, \ldots, p^{i-1}\}$ maximizing the hypervolume indicator with respect to the reference point (p^i_x, r_y) such that $|\mathrm{S}| \leq t$, where $i \in \{1, \ldots, n+1\}$ and $t \in \{0, \ldots, k\}$. Let $P(i,t)$ denote an optimal solution to subproblem (i,t) and let subproblems $(i,0)$ and $(1,t)$ be the base cases with optimal solution $P(i,0) = P(1,t) = \{\}$ (the hypervolume indicator of which is zero). For $i > 1$ and $t > 0$, let p^j be the rightmost point in an optimal subset $\mathrm{S} = P(i,t)$ to subproblem (i,t), where $j \in \{1, \ldots, i-1\}$. Then, the optimal substructure property lies in the fact that if the rightmost point is removed from S then the obtained subset $\mathrm{S} \setminus \{p^j\}$ is an optimal solution to subproblem $(j, t-1)$. The optimal solution to subproblem (i,t) is obtained from subproblems $(1, t-1), \ldots, (i-1, t-1)$ by finding a $j \in \{1, \ldots, i-1\}$ for which $H(P(j, t-1) \cup \{p^j\})$ with respect to the reference point (p^i_x, r_y) is maximized. In HypSSP, for each $t \in \{1, \ldots, k\}$, subproblems $(2,t), \ldots (n+1, t)$ are efficiently computed altogether in, an overall, amortized linear time. To this end, the value of $H(P(j, t-1) \cup \{p^j\})$, for each $j \in \{1, \ldots, n\}$, is modelled as a linear function of the reference point (x, r_y) where $x \in [p^j_x, p^{n+1}_x]$. The algorithm takes advantage of the fact that the maximum value over the set of the n linear functions forms a x-monotone polygonal chain, to compute $P(2,t), \ldots, P(n+1,t)$ in a single sweep. Since these computations are performed k times (for $t = 1, \ldots, k$), the algorithm plus the initial sorting holds $O(kn + n \log n)$. The optimal solution to the HSSP is given by the optimal solution to subproblem $(n+1, k)$, i.e., $P(n+1, k)$.

[C4] ILP Formulation [19]
Kuhn *et al.* [100] formalized the HSSP in the $d = 2$ case as an Integer Linear Programming (ILP) problem, which was later extended for the $d \geq 3$ case [99, 76]. With such a formalization, an adequate ILP solver can then be used to solve it. The

[18] An implementation is made available by the authors: `http://hpi.de/friedrich/docs/code/ssp.zip`

[19] The author of the formulation for the $d = 3$ case (Tobias Kuhn) implemented an algorithm formalizing the problem and using GLPK to solve it (which is used in the experiments of the next Chapter) however, such implementation is not available online.

ILP formulation will be explained next just for the $d = 2$ case with the purpose of providing some insight.

Let the point set $X = \{p^1, \ldots, p^n\} \subset \mathbb{R}^2$ be such that that $p^1, \ldots, p^n$ are sorted in ascending order of the x-coordinate. Consider that the area dominated by the point set X is split in $O(n^2)$ boxes, as in Figure 3.16, where the box (i, j) is the area commonly and exclusively dominated by points $p^i, \ldots, p^j$ w.r.t. X. Let the boxes (i, j) such that $i = j$ be called as outer boxes (those dominated by a single point) and as inner boxes otherwise. The area of each of these boxes is precalculated and a variable x_{ij} is associated to each of them indicating whether it is selected or not. Hence, the ILP formulation consists of maximizing the sum of the area of the selected boxes under the following restrictions: i) exactly k of the n outer boxes are selected; ii) every inner box is selected only if at least one outer box dominating it is selected. For example, in Figure 3.16, box $(2, 3)$ can only be selected if box $(2, 2)$ and/or box $(3, 3)$ is selected, i.e., if p^2 and/or p^3 is chosen.

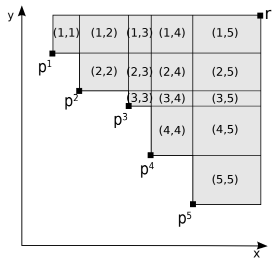


Figure 3.16: Dominated region division.

Similarly, the above ILP formulation can be extended to the $d > 2$ case [99, 76], in which case the dominated region is split in $O(n^d)$ boxes. The resulting problem has $O(n^d)$ constraints and thus, solving it requires $\Omega(n^d)$ space and time complexity.

[C5] k-Link Shortest Path Formulation (k-LSP) [20]
The k-Link Shortest Path is the problem of finding the shortest path but restricted to k-sized paths. In the formulation of the HSSPComplement for 2 dimensions as a k-link Shortest Path [100], each node represents a point of the point set $X \subset \mathbb{R}^2$ and considers two additional nodes, source and target. Consider that points $p^1, \ldots, p^n$ are sorted in ascending order of the x-coordinate and that directed arcs only exist from p^i to p^j if $i < j$. The weight of the arc between p^i and p^j is the area dominated by the set $\{p^{i+1}, \ldots, p^{j-1}\}$ which is not dominated by X, i.e., $H(\{p^{i+1}, \ldots, p^{j-1}\}, X)$. Note that the weight of the arc that goes from p^i to p^j where $j = i + 1$ is zero. Let S be the set of points in the sequence of $l - 1$ nodes in the shortest path with l links. Then, S is the optimal subset for the HSSP problem with $k = l - 1$ points. Note that the cost of feasible paths is the contribution of sets of $n - k$ to X. The HSSP problem formulated as a k-link shortest path problem, here called k-LSP, can be solved using Dynamic Programming (DP).

Kuhn *et al.* [100] proved that the digraph that results from this formulation has a property called the (concave) Monge property and so the time complexity of the DP can be improved to $O(k(n - k) + n \log n)$ time complexity (see [100] for more details). Note that $O(k(n - k) + n \log n)$ time and $O(n)$ space complexities are

[20]An implementation is made available by the authors: `https://eden.dei.uc.pt/~paquete/HSSP/hypervolume-subset.zip`

obtained if an optimal subset is not required but only its hypervolume value, while if both are required, then both complexities increase to $O(n^2)$.

The running time of k-LSP increases as k grows from 1 to $n/2$ and then decreases as k progresses from $n/2$ to n, i.e., to select t points or to remove t requires roughly the same amount of time. This is a distinct behavior from the other DP algorithms, DPHSS (see [C2]) and HypSSP (see [C3]). Unfortunately, this approach cannot be extended to $d = 3$ [100]. In comparison with HypSSP, the experimental results in [100] show that both algorithms are competitive, where HypSSP is faster than k-LSP for smaller values of k while the latter is faster for k closer to n. The turning point seems to be around $3n/5$.

[C6] Bringmann, Cabello and Emmerich algorithm for $d = 3$ (BCE3D)
Bringmann *et al.* [31] proposed an algorithm for the $d = 3$ case of HSSP that runs in $n^{O(\sqrt{k})}$ time, which will be called BCE3D here. This is the first algorithm that applies Dynamic Programming to solve HSSP in the $d = 3$ case. The idea resides in the fact that the boundary of the volume of a point set S can be described by a planar graph with $O(|\mathrm{S}|)$ vertices. This planar graph has a separator with $O(\sqrt{|\mathrm{S}|})$ vertices and the idea is to use such separator to split the problem into two subproblems. The time-complexity is related to the maximum number of partitions which is $n^{O(\sqrt{k})}$. This means that it is possible to solve the HSSP for $d = 3$ without checking every combination of k points out of the n initial points.

[C7] Branch-and-bound algorithm (B&B) [21]
Gomes [75, 76] proposed a Branch-and-Bound (B&B) algorithm to solve the HSSP for $d \geq 2$ dimensions. Gomes formulated the subproblem (A_P, R_P, N_P) as a partition of the initial point set X, where A_P, R_P and N_P represent the points already selected, the discarded ones and the ones yet to be decided upon, respectively. The first call is represented by $(\emptyset, \emptyset, \mathrm{X})$ and the branching consists of either moving a point p from N_P to A_P (i.e., select p) or moving a point p from N_P to R_P (i.e., discard p), where p is the point in N_P that contributes the most to A_P. Consequently, the first feasible solution obtained (and the first lower bound), is equal to that obtained by a(n incremental) greedy algorithm (see gHSS [D2]) and thus, has an approximation guarantee. Gomes proposed three different methods to compute bounds, one based on the greedy incremental algorithm (see [D2]), and two others based on hypervolume contributions where one relates to AllContributions problem and the other to AllContributions2 problem (see [75, 76] for more details).

The experimental results presented by Gomes [75, 76] for $d = 3, 4$ show interesting results since it outperformed solving the ILP formulation ([C4]) in the case of $d = 3$. In fact, such experiments benefited from the various algorithms proposed in Chapter 4 for updating contributions and for the greedy approximation of the HSSP. The performance of the B&B algorithm is similar to k-LSP in the sense that it is slower for k somewhere close to $n/2$ and becomes faster as k either decreases or increases.

Remarks

Table 3.4 summarizes the existent algorithms for solving HSSP exactly, indicating the number of dimensions(d), and whether or not there is an implementation avail-

[21] An implementation is made available by the author: `https://github.com/rgoomes/hssp`

Algorithm	d	d'	Time-complexity	Available	Paradigm
BF [C1]	≥ 2	≥ 4	$O(n^{d/2} \log n + n^{n-k})$	No	SDC
DPHSS [C2]	2	-	$O(kn^2)$	?	DP
HypSSP [C3]	2	2	$O(nk + n \log n)$	Yes	DP
k-LSP [C5]	2	2	$O(k(n-k) + n \log n)$	Yes	digraph-based, DP
ILP [C4]	2,3	-	$\Omega(n^d)$	Yes*	ILP
BCE3D [C6]	3	3	$n^{O(\sqrt{k})}$	No	DP
B&B [C7]	≥ 2	≥ 3	-	Yes	Branch-and-Bound

Table 3.4: Exact Algorithms for the HSSP

able online ("Available"). Column d' indicates for which number of dimensions each algorithm is the most indicated/faster.

For the 2-dimensional case, the currently fastest algorithms are HypSSP and k-LSP, the former for small values of k and the latter for k close to n. It is worth emphasizing the performance of k-LSP (and of B&B) that, unlike other algorithms whose runtime either monotonically increases or decreases as k approximates n, k-LSP is progressively slower as k approximates $n/2$ and then becomes faster as k approximates n. For the 3-dimensional case, the best alternative is to use the B&B algorithm as it was shown to be faster than the ILP solver [75, 76] and is available online. Because there are no experimental results for the BCE3D algorithm, it is not clear how the algorithm would compare against the B&B algorithm and so it is also marked in the table as one of the recommended algorithms for $d = 3$. For $d > 3$, the B&B algorithm is likely the best alternative, otherwise it would be necessary to consider all possible subsets of size $n - k$ (with BF) which can be very time consuming.

Even though the B&B algorithm is the best available alternative for $d \geq 3$, the runtime required for large n rapidly increases (for example, in Gomes [75, 76], for $d = 3$, $k = n/2$ and n close to 100, the algorithm took 100 seconds for some data sets). Therefore, B&B is recommended only for specific cases such as small n, or k close to either 1 or n. For any other cases in $d \geq 3$, if an approximation to the HSSP is enough, a greedy algorithm should be preferable.

On the extension of the $d = 2$ algorithms to $d \geq 3$ In the study of optimal distributions for the 3-dimensional case, there are up to n points that influence the optimal placement of a point and its contribution, i.e., there are up to n delimiters of the contribution of a point, while for the $d = 2$ case, there are just 2 [2]. Therefore, the choice of a point may depend on all other points. Consequently, that makes the extension of dynamic programming algorithms for the 2-dimensional case to the 3-dimensional case more difficult.

3.4.5 Greedy Algorithms for the HSSP

The greedy approximation of HSSP is an alternative to the computationally expensive exact algorithms, in particular for $d > 2$. Several greedy approaches were proposed in the literature. Most of these approaches are generic approaches that rely on hypervolume-based algorithms and thus, implementing them with different algorithms lead to different instances of such approaches. This means that the time complexity and runtime of an instance of a generic greedy approach depends on the underlying hypervolume-based algorithm used to implement it, but the approximation result does not. Four generic greedy approaches and the corresponding fastest instances are described and discussed next. Additionally, and as in the previous section, a very recent approximation algorithm to the HSSP is acknowledged at the end, but is not explained in detail.

Similarly to Section 3.4.4, for the algorithm description, let $\mathrm{X} \subset \mathbb{R}^d$ be a non-dominated point set such that $|\mathrm{X}| = n$ and that an optimal subset, $\mathrm{S}^{opt} \subseteq \mathrm{X}$, of size $k \in \{1, \ldots, n\}$ needs to be selected. Let $r \in \mathbb{R}^d$ be a reference point such that every point in X strongly dominates it.

[D1] Decremental Greedy Approach (gHSSD)
Bradstreet *et al.* [23, 21] proposed the decremental greedy approach (referred to as "Greedy Front Reduction") namely gHSSD. In gHSSD, $n - k$ points are discarded from X one at a time so as to maximize the hypervolume retained at each step. See Section 4.3 for more details on gHSSD.

The greedy decremental approach requires that, at every iteration, the contribution to X of (some) points still in X be updated. One way to implement this greedy approach is to use an algorithm to solve the LeastContributor problem at each iteration without having to keep the contributions of all points still in X updated (e.g., IWFG [B7]). Alternatively, an algorithm to compute UpdateAllContributions can be used. If none is available, an algorithm to compute AllContributions can be used instead.

For $d = 2$, the best instance of gHSSD is with UHVC2D (see [B8]) to solve UpdateAllContributions problem in $O(\log n)$ time, which leads to an instance with $O((n+(n-k)) \log n)$ time complexity. For $d \geq 3$, gHSSD has to be implemented based on algorithms for the AllContributions problem. For $d = 3$, the best instance has $O((n - k)n \log n)$ time complexity by using EF algorithm (see [B2]) while for $d > 3$, the best instance has $O(kn^{d/2} \log n)$ time complexity by using BF1 algorithm (see [B1]). All of the above gHSSD instances only require $O(n)$-space complexity even when the points in the optimal greedy subset are sought.

Bader and Zitzler [10] proposed a "one shot" version of gHSSD that computes all contributions and then removes at once the $n-k$ points that contribute the least. This version leads to poorer approximations and is thus recommended only when updating contributions is computationally expensive, i.e., for high values of d.

Regarding theoretical results, the currently known results on gHSSD are not very optimistic. Let S^g be the (decremental) greedy solution given $\mathrm{X} \subset \mathbb{R}^d$ and $k \in \{1, \ldots, |\mathrm{X}|\}$. Bringmann and Friedrich [35] showed that the approximation ratio to the HSSPComplement problem, i.e., the ratio $(H(\mathrm{X})-H(\mathrm{S}^g))/(H(\mathrm{X})-H(\mathrm{S}^{opt})$, can be arbitrarily large, even though in many cases it is one or close to one. In particular, they showed that there is a set of points $\mathrm{X} \subset \mathbb{R}^d$ such that $|\mathrm{X}| = n \geq 4$ and $d \geq 3$ for which this ratio is higher or equal to $\kappa \geq 1$ for $k = n - d, \ldots, n - 2$. That is the case of the set $\mathrm{X} = \{(1 + \epsilon, 1 + \epsilon, 1 + \epsilon), (1 + \epsilon + \sigma, 1, 1), (1, 1 + \epsilon + \sigma, 1), (1, 1, 1 + \epsilon + \sigma)\}$, considering maximization and the reference point $(0, 0, 0)$, where $\epsilon = \frac{1}{2\kappa d^2}$ and $\sigma = d^2\epsilon^2$.

[D2] Incremental Greedy Approach (gHSS)
Bradstreet *et al.* [23] also proposed the greedy incremental approach (referred to as "Greedy Front Addition") which will be called here as gHSS. In gHSS, k points are selected from X one at a time so as to maximize the hypervolume gained at each step. See Section 4.4 for more details on gHSS.

The greedy incremental approach requires that, at every iteration, the contribution to S of (some) points still in X have to be updated, where S is the set of already selected points and $|S| \leq k$. Thus, gHSS can be computed as a sequence of UpdateAllContributions2 problems. However, due to the absence of dedicated algorithms for Problems 3.8 and 3.4, and because algorithms specific for Problems 3.7 and 3.3 cannot be straightforwardly applied to solve them, algorithms for the OneContribution problem are currently the best option for computing gHSS.

For the general $d \geq 2$ case, one option is to instantiate gHSS with IHSO (see [B4]) in order to compute the required contributions, one by one. For $d = 3$, the best alternative is to use UHV3D (see [B5]) to compute each contribution in linear time. Such an instance of gHSS would have $O(k^2 n)$ time complexity by noting that the contribution of every point $p \in X$ to S (i.e., $H(p, S)$) has to be computed, at most, k times and, each time, $|S|$ has no more than k points. Similarly, HV4D (see [A6]) could be used for $d = 4$, leading to $O(k^3 n)$ time complexity. For $d > 4$, the best alternative is to either iterate over the version of WFG (see [A8]) adapted to compute OneContribution (see "Remarks" in Section 3.4.3), or over Chan's algorithm (see [A9]) to compute Hypervolume.

Regarding theoretical results, the fact that the HSSP consists of maximizing a submodular function subject to a cardinality constraint automatically provides an approximation guarantee of $(1 - 1/e)$ with the incremental greedy approach (see Section 3.2). Let S^g be the (incremental) greedy solution, then approximation ratio to the HSSP problem, i.e., the ratio $H(S^g)/H(S^{opt})$ is, at least, $(1 - 1/e)$.

[D3] Local Search (LS)
Bradstreet *et al.* [21, 23] also proposed a Local Search (LS) Approach based on the premise that for large n and small values of k it may be advantageous to try out several sets of size k, i.e., perform several cheap computations instead of the more demanding ones of updating all contributions (e.g., as in gHSSD). In LS, a k-size subset S of X is initially selected randomly. Then, a small number of points is (randomly) replaced by others in $X \setminus S$ and the new subset is accepted if the hypervolume indicator has improved. This replacing step is repeated until a time limit is exceeded. Basseur *et al.* [12] proposed two similar approaches, one called the first-improvement hill-climbing local search (here referred to as FHLS) and another called Greedy Sequential Insertion (GSI). As LS, both of them randomly select a k-sized point set, but the replacement step is more restricted as they only swap a single point by another. In the case of FHLS, the main difference to LS, is that FHLS is more exhaustive as it only stops when no swap leads to an improvement. In the case of GSI, it tests the swapping in of each point left out just once, and the swap is done in a greedy way by adding a new point to the set and excluding the least contributor.

In [21], LS was compared against the gHSSD (combined with HSO) and it was concluded that the former was able to find subsets with equal or better hypervolume in less time than those found by the latter, when $k < n/2$. However, later in [23], with a faster implementation of gHSSD and the new gHSS (both combined with IHSO), LS performed worse when given the same (and even twice the) time taken by gHSSD and gHSS. Nevertheless, with the current state-of-the-art where algorithms such as HV4D and WFG could make LS and gHSS faster and algorithms as EF, BF and IWFG could make gHSSD faster, it is not clear which greedy approach could be now more advantageous, which would provide better solutions in less amount of time. Moreover, note that the time complexity of LS-based algorithms depend not only on the hypervolume-based algorithms used to evaluate the k-sized subsets, but also on the number of times such algorithms are called. For example, in FHLS, $\Omega(k(n-k))$ subsets are evaluated, i.e., the HYPERVOLUME problem is solved $\Omega(k(n-k))$ times. In the case of GSI, the LEASTCONTRIBUTOR problem is computed $n-k$ times.

[D4] Global Simple Evolutionary Multiobjective Optimizer (GSEMO) Friedrich and Neumann [71] proposed GSEMO, a multiobjective evolutionary algorithm that is expected to achieve a $(1-1/e)$-approximation to the HSSP in $O(n^2(\log n+k))$ iterations. In GSEMO, each individual in the population encodes a subset S of X, which represents a solution to the HSSP. GSEMO considers the (maximization) of a bi-objective problem, where the first objective is the hypervolume indicator of the encoded set S if $|\mathrm{S}| \leq k$ and is -1 otherwise, while the second objective is the number of points left out of S, i.e., $|\mathrm{X} \setminus \mathrm{S}|$.

In GSEMO, the population is initialized with a single randomly generated individual. In each iteration of GSEMO, an individual from the population, representing a subset S, is chosen uniformly at random and each point is added/removed to/from S with probability $1/n$. The new individual is inserted in the population only if it is not weakly dominated and if so, all individuals in the population it dominates are discarded. Note that the population will keep at most $n+1$ solutions as the second objective can only have values $0,\ldots,n$ and only the solution with highest hypervolume indicator for each of these values is kept.

In short, in GSEMO, $O(n^2(\log n+k))$ different subsets have to be evaluated until a subset of size k that is an $(1-1/e)$-approximation is expected to be found. Therefore, the time complexity of the overall algorithm is the number of iterations/evaluations multiplied by the time complexity needed to compute the hypervolume indicator of each solution and the best one is obtained with Chan's algorithm (see [A9]).

[D5] Efficient Polynomial-time Approximation Scheme (EPTAS) Very recently, Bringmann *et al.* [31] proposed an algorithm based on Dynamic Programming to approximate the HSSP for any constant number of dimensions d, that runs in $O(n\epsilon^{-d}(\log n + k + 2^{O(\epsilon^{-2}\log 1/\epsilon)^d}))$ and has an approximation guarantee of $O(1-\epsilon)$ to the optimal solution. The algorithm provides a subset S of size at most k with the approximation guarantee, but does not provide the exact value of $H(\mathrm{S})$ only an approximation. To compute it exactly, an algorithm as the ones described in Section 3.4.2 should be used, which would increase the time complexity of $O(n^2)$ of EPTAS when $d \geq 6$.

Remarks

The most advantageous combination of each greedy approach with state-of-the-art hypervolume-based algorithms, for each number of dimensions, are shown in Table 3.5. This list includes the combinations that lead to algorithms with the best time complexity and includes the currently fastest alternatives than can be constructed from implementations found online. In the case of the time-complexity analysis of LS, t stands for the number of k-sized subsets evaluated. Hypervolume-based algorithms with available implementations but that are not available in combination with the greedy approach, and yet should be easily integrated, have *Availability* marked as "Yes*". Note that in the case of $d=2$, there are very efficient algorithms

Algorithm	Auxiliar Alg.	d	d'	Time-complexity	Avail.
gHSSD [D1]	UHVC2D [B8]	2	2	$O((n-k)\log n)$	Yes*
	EF [B2]	3	3	$O((n-k)n\log n)$	Yes*
	BF1 [B1]	≥ 2	≥ 4	$O((n-k)n^{d/2}\log n)$	No
	IWFG [B7]	≥ 2	≥ 4	-	Yes
	WFG [A8]	≥ 2	≥ 4	$O((n-k)n^{d-1})$	Yes*
gHSS [D2]	UHV3D [B5]	3	3	$O(k^2n)$	Yes*
	HV4D [A6]	4	4	$O(k^3n)$	Yes*
	WFG [A8]	≥ 2	≥ 5	$O(k^dn)$	Yes*
	Chan [A9]	≥ 2	≥ 4	$O(nk^{(d+3)/3}\text{polylog } k)$	No
LS [D3]	HV3D [A4]	$2, 3$	$2, 3$	$O(tk\log k))$	Yes*
	HV4D [A6]	4	4	$O(tk^2)$	Yes*
	WFG [A8]	≥ 2	≥ 5	$O(tk^{d-1})$	Yes*
	Chan [A9]	≥ 2	≥ 4	$O(tk^{d/3}\text{polylog } k))$	No
GSEMO [D4]	HV3D [A4]	$2, 3$	$2, 3$	$O(n^3(\log n + k))$	Yes*
	HV4D [A6]	4	4	$O(n^4(\log n + k))$	Yes*
	WFG [A8]	≥ 2	≥ 5	-	Yes*
	Chan [A9]	≥ 2	≥ 4	$O(n^{(d+6)/3}\text{polylog } n(\log n + k))$	No
EPTAS [D5]		≥ 2	≥ 2	$O(n^2)$	No

Table 3.5: Greedy Algorithms for the HSSP.

to compute HSSP exactly (see Section 3.4.4) and thus, it is questionable whether a greedy algorithm is useful in this case. Anyway, this case is also considered in this summary for completeness.

For the greedy decremental approach, UHVC2D and EF are the recommended algorithms for $d = 2$ and $d = 3$, respectively. For the remaining ones, BF could be used but since it is not available online, the alternative is to use either IWFG or WFG. Due to the lack of experimental results comparing IWFG and WFG (and BF1), it is not clear which is the best one in practice. For the remaining greedy approaches, for $d = 3$ and $d = 4$, the recommended algorithms are HV3D and HV4D, respectively. For $d > 4$, WFG is the best alternative. Chan's algorithm has better time complexity than HV4D and WFG, however due to the absence of comparison studies and of an implementation it is not clear whether it would perform faster.

Currently, only gHSS and EPTAS provide approximation quality. However, the best one cannot be decided based on these guarantees as these only provide a lower bound and do not imply which algorithm performs better in practice. Moreover, a lack of an approximation guarantee does not mean either that an algorithm cannot perform better in practice than those that have it.

Finally, when comparing generic greedy algorithms such as gHSSD, gHSS, LS and GSEMO, it is important to have in mind that the runtime required by them relies on an algorithm for a hypervolume-based problem and the choice of such an algorithm does not impact the approximation quality. Thus, it is important to have studies comparing the quality of the approximation independently of the algorithms' runtime as better algorithms for hypervolume-based problems may be developed in the future which would lead to faster instances of such generic greedy algorithms.

3.5 Data Sets

To test the performance of the algorithms for hypervolume-related problems, data sets representing different front shapes are typically used. The data sets[22] listed in this section are the ones relevant for this thesis and include some of the most used, and new ones as well. As these data sets are used in the next two chapters, this section also includes a description of how they were generated.

Every point p in a n-sized point set $\mathrm{X} \subset [0,1]^d$ was generated in the following way, depending on the front that X represents:

Linear:

$$p_i = \frac{|X_i|}{\sum_{j=1}^{d} |X_j|}, \ X_i \sim Uniform(0,1), \text{ for } i = 1,..,d$$

(Spherical) Convex:

$$p_i = 1 - |X_i|/||X||, \ X_i \sim Normal(0,1), \text{ for } i = 1,..,d$$

(Spherical) Concave:

$$p_i = |X_i|/||X||, \ X_i \sim Normal(0,1), \text{ for } i = 1,..,d$$

Wave-w ($d = 2$):

$$\begin{aligned}
&p_1 = X_1 \cos\left(-\frac{\pi}{4}\right) - Y_1 \sin\left(-\frac{\pi}{4}\right), \quad p_2 = X_1 \sin\left(-\frac{\pi}{4}\right) + Y_1 \cos\left(-\frac{\pi}{4}\right) + 1,\\
&X_1 = \sqrt{2} Z_1, \quad Y_1 = r \cdot \cos(2w\pi Z_1) - r,\\
&Z_1 \sim Uniform(0,1), \quad r = \frac{0.2}{w}
\end{aligned}$$

Degenerate ($d = 3$):

$$\begin{aligned}
&p_1 = \cos\left(\frac{\pi}{2}X\right)\cos\left(\frac{\pi}{4}\right), \quad p_2 = \cos\left(\frac{\pi}{2}X\right)\sin\left(\frac{\pi}{4}\right), \quad p_3 = \sin\left(\frac{\pi}{2}X\right),\\
&X \sim Normal(0,1)
\end{aligned}$$

Cliff ($d = 3$):

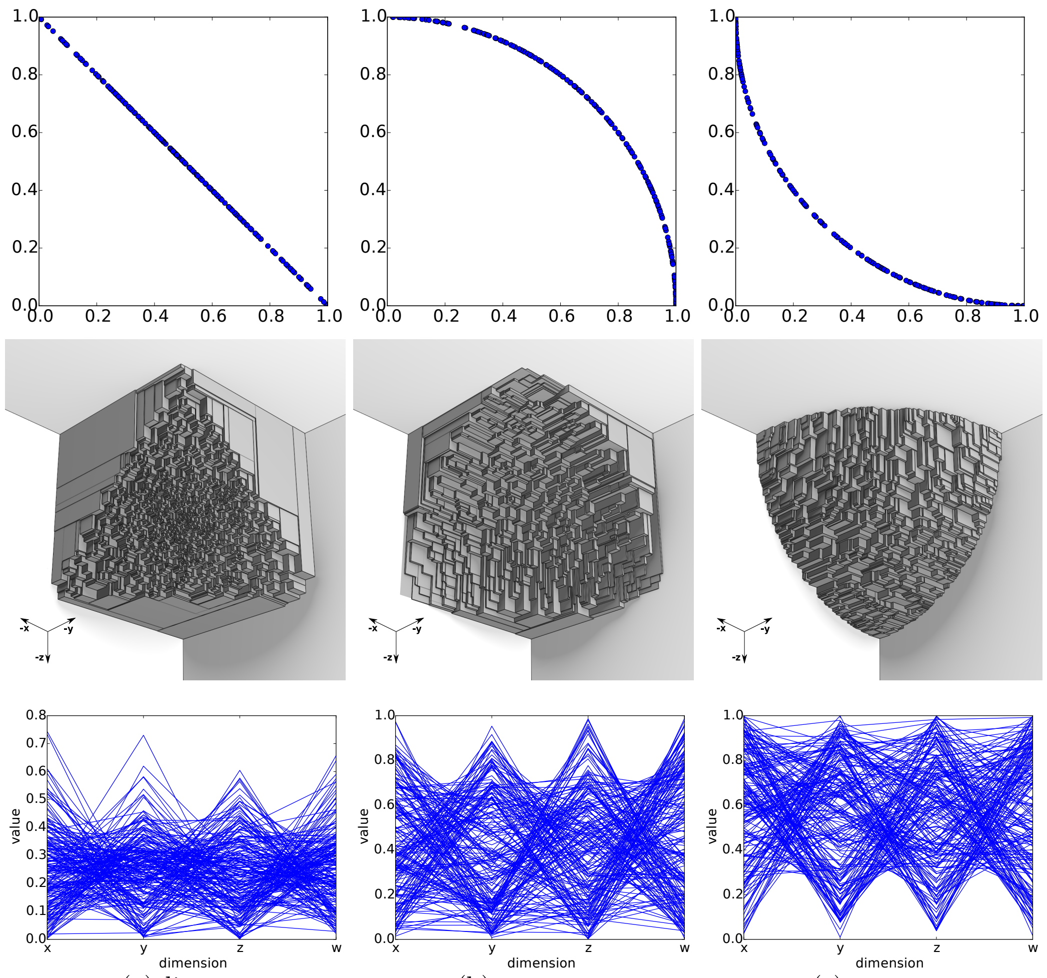
$$\begin{aligned}
&p_i = 1 - |X_i|/\|X\|, \ X_i \sim Normal(0,1), \text{ for } i = 1,2\\
&p_3 \sim Uniform(0,1)
\end{aligned}$$

Cliff ($d = 4$):

$$\begin{aligned}
&p_i = 1 - |X_i|/\|X\|, \quad X_i \sim Normal(0,1) \text{ for } i = 1,2\\
&p_{j+2} = 1 - |Y_j|/\|Y\|, \ Y_j \sim Normal(0,1) \text{ for } j = 1,2
\end{aligned}$$

Hard ($d = 4$ and n is even):

$$\begin{aligned}
&p^j = \left(\frac{n+2j}{2n}, \frac{n-j-1}{n}, \frac{j}{n}, \frac{n-2j-2}{2n}\right) \quad \text{for } j = 0, \ldots, \frac{n}{2} - 1\\
&p^j = (p_w^l, p_z^l, p_y^l, p_x^l) \quad \text{for } j = \frac{n}{2}, \ldots, n-1 \text{ where } l = j - \frac{n}{2}
\end{aligned}$$



(a) linear (b) concave (c) convex

Figure 3.17: Data sets: linear (left), concave (middle) and convex (right) in two, three and four dimensions in first to third row, respectively.

Some data sets are specific to a given number of dimensions d which is indicated right after the data set name. Figures 3.17 and 3.18 illustrate the data sets. Most of the data sets are known from the literature, namely, linear, convex and concave are the most common [54, 129, 61], the degenerate front is the Pareto front for the DTLZ5 test problem [54], cliff (in $d = 3$) is a data set where all points are nondominated in their projection onto the (x, y)-plane [61] and the hard data set [102] [23] was constructed as a difficult data set for WFG algorithm (see [A8]). Additionally, an extended version of cliff to $d = 4$ is described, and so is a mixed convex/concave front, so that not all data sets represent either fully convex or fully concave fronts.

Apart from the *Hard* data set, for every data set and number of dimensions (2, 3 or 4), 10^5 points were randomly generated. All smaller sets of points where sampled from the initial 10^5 points. For each size n, a set of n points was created

[22]A script to generate most of the data sets used is available at: `https://github.com/apguerreiro/DataSetGenerator`

[23]The description of the hard data set presented here is a simplified version for $d = 4$ of the one proposed by Lacour *et al.* [102], where the relative position between points is maintained but not the absolute values of their coordinates. A script to generate the original hard data set is made available by the author at: `https://github.com/renaudlr/moo-nondominated-sets`

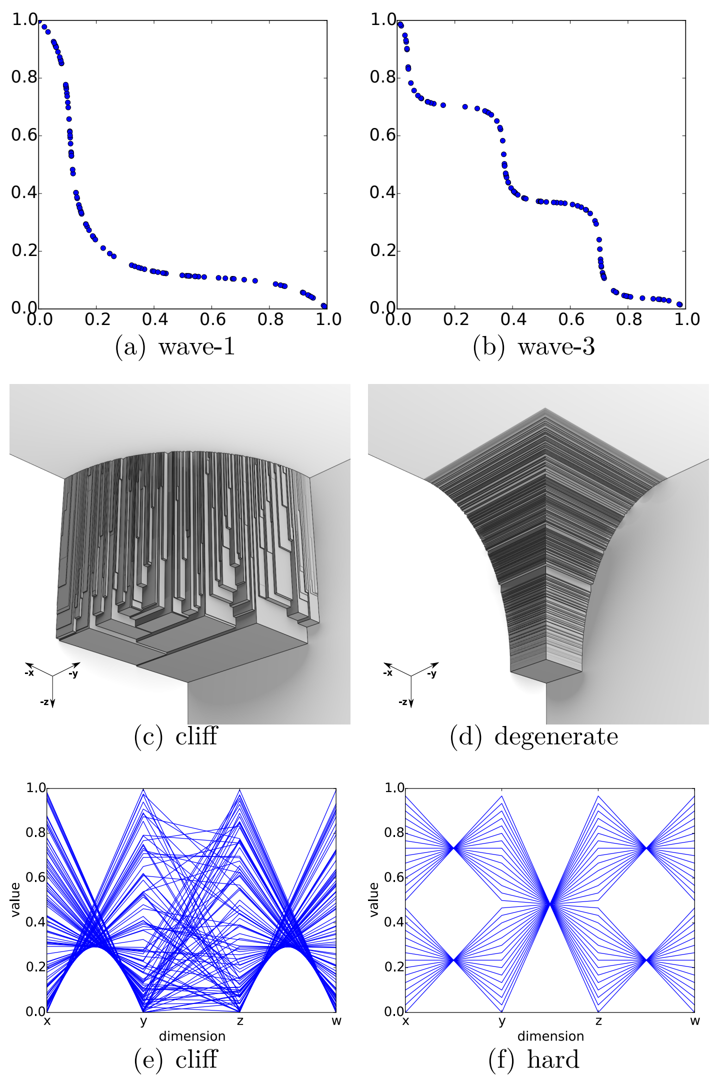


Figure 3.18: Data sets: wave-1 and wave-3 in $d = 2$ (top figures), cliff (middle left) and degenerate (middle right) in $d = 3$, and cliff (bottom left) and hard data set (bottom right) in $d = 4$.

by randomly selecting n points from the initial set. The Hard data set (for $d = 4$) was the only where points were (deterministically) generated as described for each size n. Moreover, all data sets are normalized to the range $[0, 1]$.

3.6 Concluding Remarks

In this chapter, the properties of the hypervolume indicator were reviewed, namely, strict monotonicity, scaling independence, optimal μ-distributions, and submodularity. These properties tell us what sets the indicator prefers, i.e., which sets the indicator is expected to benefit (closer to the Pareto front and well spread, but depending on the slope of the front). The properties of archiving algorithms based on the hypervolume were also reviewed, which indicate what to expect from the outcome of Archiving Algorithms that perform environmental selection based on the indicator. Moreover, given the characteristics of the archivers, some settings can be recommended, e.g., if the archiver uses an exact algorithm for the HSSP, then the strategy that should provide better results is the $(\mu + \mu)$ strategy. All of these properties reinforce the interest in hypervolume-based selection in EMOAs.

The last sections reviewed the existing algorithms for hypervolume-based problems relevant to EMOAs. Although there has been a significant improvement in the computation of the hypervolume indicator over the last years, the same cannot be said for the remaining problems. The use of a particular algorithm for computing hypervolume contributions and the HSSP is still limited. In particular, the use of hypervolume-based selection in EMOAs is still limited in the many-objective case ($d > 4$), but even the low dimensional cases would benefit greatly from more efficient algorithms with fast implementations. Furthermore, without knowing how to solve the low dimensional cases very efficiently, there is not much hope of having fast algorithms for many-objective cases.

The improvements on algorithms for the hypervolume indicator may be relevant and extendable for the computation of contributions due to the close relation between the two problems. For the HSSP, the most recent algorithms for $d = 2, 3$ show that it is possible to compute HSSP exactly without testing every possible combination of k-sized subsets from a set of n points. However, even for $d = 3$ the existing algorithms still do not seem to be efficient/fast enough unless k and/or n are small. The alternative may be to use very fast but less precise algorithms. Therefore, investing in greedy algorithms and in understanding their guarantees/properties may result in interesting alternatives.

Overall, the interest in hypervolume-based EMOAs, the theory supporting the hypervolume indicator and the difficulties faced in the computation of hypervolume-related problems were highlighted in this chapter. These aspects motivate the interest of this thesis in the indicator, and justify the investment in faster algorithms for hypervolume-related problems, which are presented in the next chapter.

Chapter 4

Hypervolume Subset Selection

The theoretical advantage of the hypervolume indicator and of performing subset selection based on such an indicator in the context of Evolutionary Algorithms were exposed in the previous Chapter 3, as well as the drawbacks concerning computational costs. These drawbacks impose a limit to hypervolume-based EMOAs on the number of dimensions considered, number of generations, the population size and/or number of offspring. Selection in EMOAs based on the exact HSSP has been restricted mainly to $k = n - 1$ and to $d = 2, 3$. Greedy approaches have been proposed although not much is known about their approximation guarantees.

This chapter focuses on three main goals. The first goal is to significantly reduce the computational burden of hypervolume-based subset selection in EAs for low dimensional problems. This is achieved with new algorithms to compute/update hypervolume contributions, which lead to a faster computation of the HSSP given $k = n-1$, and to faster (decremental/incremental) greedy algorithms for the HSSP in $d = 3, 4$. The efficient update of hypervolume contributions at the core of these algorithms is also useful for steady-state EMOAs. The second goal is to extend the knowledge on the theoretical properties of the above greedy algorithms. Finally, the third goal is to provide insight on the performance of hypervolume-based subset selection, particularly in the context of evolutionary algorithms and to bridge practical and theoretical results. This is achieved through extensive experimental studies on the approximation quality of such greedy algorithms for one-time subset selection as well as on the approximation quality of archiving algorithms (for sequential subset selection) based on exact and on decremental/incremental greedy algorithms.

This chapter is organized as follows. Section 4.1 describes new algorithms to compute and update the hypervolume indicator in $d = 3, 4$, which are then extended in Section 4.2 to compute hypervolume contributions. Section 4.3 is dedicated to the decremental greedy approximation of the HSSP where new algorithms are proposed for $d = 3, 4$ by making use of the algorithms proposed in the previous section, and an approximation bound is derived. Section 4.4 is dedicated to the incremental greedy approximation of the HSSP where new algorithms for $d = 2, 3$ are proposed, and it is shown that there is no approximation bound for the HSSPComplement using this approach. The greedy algorithms are then experimentally evaluated in Section 4.5 for one-time subset selection. In Section 4.6, the theoretical and practical aspects of using such greedy approaches in archiving algorithms are addressed by providing experimental results on their performance. Finally, the chapter's concluding remarks are drawn in Section 4.7.

4.1 Computation of the Hypervolume Indicator

In this section, HV3D (see [A4]) is modified in order to allow for incremental and decremental updates in linear time, resulting in a new algorithm that will be called HV3D$^+$. This is achieved by preprocessing the input points and setting up a data structure to support the subsequent hypervolume calculation. Hypervolume updates are performed by updating the data structure to reflect the insertion or the removal of a point and either recomputing the new hypervolume as a whole or computing the corresponding contribution. By iterating over such updates in three dimensions, a new $O(n^2)$-time algorithm for four dimensions is obtained as an alternative to HV4D (see [A6]).

The point set $\{s^1, \dots, s^{14}\}$ in the example of Figure 4.1 will be used in this and the following section, mostly to explain the computation of the contribution of $p = s^{10}$ to that set in (x, y, z)-space and in the (x, y)-plane at $z = p_z$. Therefore, the contribution of p is highlighted and the hypervolume of the points below p in z-coordinate are shed gray in the projections onto the (x, y)-plane.

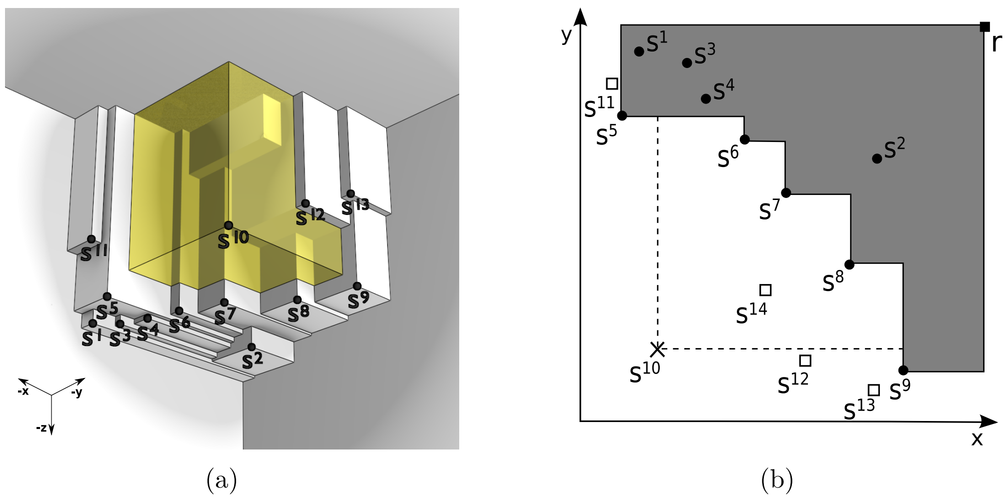


Figure 4.1: An example in three-dimensions where $s_z^1 < \cdots < s_z^{14}$ and $p = s^{10}$. All points are nondominated except s^{14}, which is dominated by s^{10}.

4.1.1 Three Dimensions

Data Structures

Maintaining the set of points visited so far whose projections on the (x, y)-plane are nondominated and being able to access them in ascending order of the y coordinate are key aspects of both HV3D and HV3D$^+$. Let S represent a data structure for that purpose, which can be either a balanced binary tree, or a linked list. Note that, since S* contains nondominated points only, ascending order of coordinate y is equivalent to descending order of coordinate x.

Consider the following operations on S, as well as the corresponding operations obtained by switching the roles of the x and y coordinates. It is assumed that $s \in$ S, $p \in \mathbb{R}^3$, and $q <_L p$ for all $q \in$ S, where $q <_L p$ denotes that q is lexicographic less than p in dimensions z, y, x. Thus, $q_z \leq p_z$.

Algorithm 4.1 genericSweep$_y(p, \mathrm{S}, s)$

```
Require: $s = \mathsf{outerDelimiter}_x(p, \mathrm{S})$
1: $e \leftarrow \mathsf{next}_y(s, \mathrm{S})$
2: procedureA$(\mathrm{S}, p, s, e)$
3: while $p_x \leq e_x$ do
4:     $q \leftarrow e$
5:     $e \leftarrow \mathsf{next}_y(e)$
6:     procedureB$(\mathrm{S}, p, q, e)$
7: procedureC$(\mathrm{S}, p, q, e)$
8: return  *info*
```

head$_y(\mathrm{S})$ Return the point $q \in \mathrm{S}$ with the smallest q_y.

next$_y(s, \mathrm{S})$ Return $q \in \mathrm{S}$ with the smallest $q_y > s_y$.

remove(s, S) Remove s from S.

outerDelimiter$_y(p, \mathrm{S})$ Return the point $q \in \mathrm{S}$ with the smallest $q_y > p_y$ such that $q_x < p_x$.

removeDominated$_y(p, \mathrm{S}, s)$ If $s = \mathsf{outerDelimiter}_x(p, \mathrm{S})$, remove all points $q \in \mathrm{S}$ such that $p^* \leq q^*$ from S, and return them sorted in ascending order of q_y.

computeArea$_y(p, \mathrm{S}, s)$ If $s = \mathsf{outerDelimiter}_x(p, \mathrm{S})$, compute and return the area of the region exclusively dominated by p^* with respect to S^*.

add$_y(p, \mathrm{S}, s)$ Insert p into S if $s_y < p_y < \mathsf{next}_y(s, \mathrm{S})$ or $p_y < \mathsf{head}_y(\mathrm{S})$. In the latter case, s should be NULL.

Operation outerDelimiter requires time in $O(\log n)$ if S is a binary tree and in $O(n)$ if it is a linked list. Operations head, next, add and remove take $O(1)$ time on a linked list (because $s \in \mathrm{S}$) and, in general, $O(\log n)$ time on a balanced binary tree. On a tree, head can also be implemented in constant time just by caching a pointer to the head node.

Operations removeDominated and computeArea both follow the template presented in Algorithm 4.1. Points $q \in \mathrm{S}$ whose projections are dominated by p^* (inner delimiters) are visited in ascending order of q_y by starting at $s = \mathsf{outerDelimiter}_x(p, \mathrm{S})$, which must be passed as an input argument, and stopping at the first subsequent point e such that $p^* \not\leq e^*$. Routines procedureA, procedureB and procedureC, respectively, represent the pre-processing, processing and post-processing operations associated with the sequence of points visited. In operation removeDominated, an empty point list is initialized in procedureA. In procedureB, the visited points q are added to that list, and are removed from S by invoking remove(q, S), whereas procedureC does nothing. The list is returned as *info*. In operation computeArea, the area of the rectangle $[(p_x, p_y), (s_x, e_y)]$ is computed, and is stored in *info* in procedureA. Similarly, the area of $[(p_x, q_y), (q_x, e_y)]$ is computed and added to *info* in procedureB. As before, procedureC does nothing, but this routine will become important later, in Section 4.2.

Returning to the example of Figure 4.2(a) given p, $\mathrm{S} = \{s^5, \dots, s^9\}$, and $s = s^9$, the delimiters of p^* are visited starting at s^9 and stopping at s^5. Operation

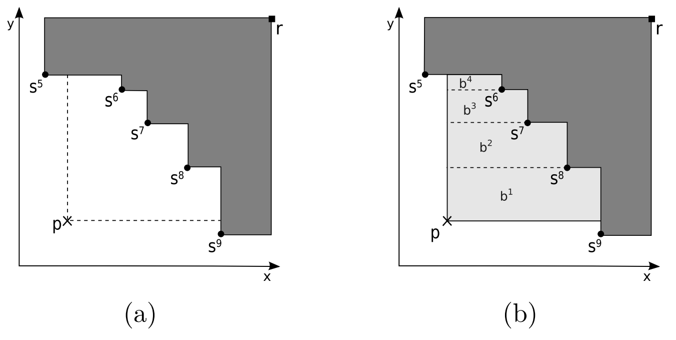


Figure 4.2: Computing the contribution of p^* to T*.

removeDominated removes points s^8, s^7 and s^6 from S, and returns them in this order in a list, whereas computeArea returns the area exclusively dominated by p^*, and leaves S unmodified. To compute this area, computeArea divides it in axis-parallel boxes as depicted in the example of Figure 4.2(b). procedureA computes the area of box b^1, while boxes b^2, b^3 and b^4 are computed in this order, each one in a call to procedureB. Note that this is an alternative to the method used in HV3D (see [A4]) to compute the contribution of p^*. In this case it is not necessary to remove dominated points beforehand and this will be an important aspect further ahead to the computation and update of contributions (see Section 4.2). Since the processing of each point in removeDominated and computeArea is dominated by the complexity of add, next and remove, the resulting time complexity upper bounds are $O(t)$ on lists and $O(t \log |\mathrm{S}|)$ on binary trees, where t denotes the number of inner delimiters of p^*.

Preprocessing

Algorithm 4.2 reproduces the sequence of binary-tree operations performed in HV3D. Input points are stored and visited in ascending lexicographic order of coordinates z, y and x to ensure data-structure consistency and well-defined operations in the presence of repeated z coordinates. For each $p \in \mathrm{Q}$, its rightmost outer delimiter, s, is looked up in binary tree T (line 4). Then, the points in T whose projections are dominated by p^* are removed (line 5), and p is added to T (line 8). Pointers to the outer delimiters of p^* in T* are saved as attributes of p ($p.cx$ and $p.cy$ in lines 6 and 7) for future use. Sentinel nodes in T guarantee that such outer delimiters always exist.

As in HV3D, each input point is visited at most twice, once when it is added to T, and again if it has to be removed from T, at a cost of $O(\log n)$ time in both cases. Determining and saving the outer delimiters of each point (lines 4, 6 and 7) also takes $O(\log n)$ time per input point. Therefore, this preprocessing is performed in $O(n \log n)$ time.

Note that, although the input set X is required to be a nondominated point set, dominated points in Q can be easily detected and discarded at no extra cost by checking immediately after line 4 whether $\mathsf{next}_y(s, \mathrm{S})$ dominates p, and skipping to the next input point if it does.

Algorithm 4.2 HV3D$^+$ – Preprocessing

```
Require: $X \subset \mathbb{R}^3$ // a set of $n$ nondominated points
Require: $r \in \mathbb{R}^3$ // the reference point
1: $Q \leftarrow X$ // linked list sorted in ascending lexicographic order of coordinates $z$, $y$, $x$
2: $T \leftarrow \{(r_x, -\infty, -\infty), (-\infty, r_y, -\infty)\}$ // binary tree sorted in ascending order of dimension $y$
3: for each $p \in Q$ do
4:     $s \leftarrow \mathsf{outerDelimiter}_x(p, T)$
5:     $\mathsf{removeDominated}_y(p, T, s)$
6:     $p.cx \leftarrow s$
7:     $p.cy \leftarrow \mathsf{next}_y(s, T)$ // same as $\mathsf{outerDelimiter}_y(p, T)$
8:     $\mathsf{add}_y(p, T, s)$
9: return Q
```

Hypervolume Computation

Algorithm 4.3 can be seen as a reimplementation of HV3D using a linked list, L, instead of a binary tree, T, and follows the same structure as Algorithm 4.2. However, the outer delimiters of each input point are now required to be known in advance as a result of preprocessing. This allows all next, add and remove operations to be implemented in constant time, as explained earlier in the "Data Structures" subsection.

The hypervolume indicator is computed in a similar way to HV3D. Variables *area* and *vol* are used to store, respectively, the area dominated by the points in L^* and the volume of the region dominated by the points visited so far up to the current point, p. The volume is accumulated in *vol* at the beginning of the loop by multiplying the current (base) area by the height of the current slice (line 4). The area dominated by the points in L^* is updated by adding the contribution $H(p^*, L^*)$ to the current area (line 6). Finally, the total volume is updated with the volume of the last slice (line 10), and returned.

Since all inner delimiters visited in computeArea are immediately removed in removeDominated, and the complexity of both of these functions is now linear in the number of such delimiters, the complexity of Algorithm 4.3 amortizes to $O(n)$.

Data Structure Updates

Adding a new point u to the data structure maintained by HV3D$^+$ requires setting attributes $u.cx$ and $u.cy$, updating the corresponding attributes of the remaining points in the lexicographically sorted list Q, and inserting u into Q. These operations are performed in linear time in a single sweep of Q, as follows (cy attributes are updated in a similar way, but with the roles of the x and y coordinates switched):

- Set $u.cx$ to the point $q \in Q$ with the smallest $q_x > u_x$ such that $q_y < u_y$ and $q <_L u$. If such a point is not unique, the alternative with the smallest q_y is preferred.

- For $q \in Q$, set $q.cx$ to u iff $u_y < q_y$ and $u <_L q$, and either $q_x < u_x < (q.cx)_x$ or $u_x = (q.cx)_x$ and $u_y \leq (q.cx)_y$.

Algorithm 4.3 HV3D$^+$ – Hypervolume computation

```
Require: $X \subset \mathbb{R}^3$ // a set of $n$ nondominated points
Require: $r \in \mathbb{R}^3$ // the reference point
Require: Q // a linked list containing X sorted in ascending order of dimension $z$ with
    $p.cx$ and $p.cy$ set for all $p \in Q$
 1: $L \leftarrow \{(r_x, -\infty, -\infty), (-\infty, r_y, -\infty)\}$ // linked list sorted in ascending order of
    dimension $y$
 2: $vol, area, z \leftarrow 0$
 3: for each $p \in Q$ do
 4:     $vol \leftarrow vol + area \cdot (p_z - z)$
 5:     $s \leftarrow p.cx$
 6:     $area \leftarrow area + \mathsf{computeArea}_y(p, L, s)$
 7:     $\mathsf{removeDominated}_y(p, L, s)$
 8:     $\mathsf{add}_y(p, L, s)$
 9:     $z \leftarrow p_z$
10: $vol \leftarrow vol + area \cdot (r_z - z)$
11: return $vol$ // $H(X)$
```

- Insert u into Q immediately before the point $q \in Q$ with the lexicographically smallest q such that $u <_L q$.

As an example, let s^{10} in Figure 4.1 (b) be the new point u to be inserted, and Q contain all of the remaining points. Then, $s^{10}.cx$ is set to s^9 and $s^{10}.cy$ is set to s^5. Also, $s^{12}.cy$, which is s^7 before s^{10} is inserted, is set to s^{10}.

Although one may require that $Q \cup \{u\}$ be a nondominated point set, handling dominated points arising from the insertion of u is simple. If u is dominated by points in Q, which can be checked in constant time per point while sweeping Q, then u is simply discarded. If some points in Q are dominated by u, they will not be referenced as delimiters (*cx* or *cy*) of any nondominated point in $Q \cup \{u\}$. This is because any references to points dominated by u will either be made by other points dominated by u or have been updated to refer to u itself. Such dominated points can either be simply removed from Q or be marked as such and remain in Q as it will be the case in Section 4.2.

Removing a point $u \in Q$ also requires updating the *cx* and *cy* attributes of the remaining points, as follows:

- For every $p \in Q \setminus \{u\}$ such that $p.cx = u$, set $p.cx$ to the point $q \in Q \setminus \{u\}$ with the smallest $q_x > p_x$ such that $q_y < p_y$ and $q <_L p$ (and analogously for $p.cy$). If such a point is not unique, the alternative with the smallest q_y (respectively, q_x) is preferred.

Assuming that Q does not contain any dominated points, this is also achieved in linear time by performing essentially the same sequence of operations in Algorithm 4.2, but using a linked list, L, instead of binary tree, T, and replacing the call to $\mathsf{outerDelimiter}_x$ by $p.cx$. In addition, if $p.cx = u$, variable s should be set to $p.cy$ instead, and the roles of dimensions x and y should be reversed in lines 5 to 8, for that iteration.

In the example of Figure 4.1 (b), let s^{10} be the point u to be removed from Q, and $p = s^{12}$ be the current point. Hence, $s^{12}.cx = s^9$, $s^{12}.cy = s^{10}$, and L =

$\{s^{11}, s^5, s^6, s^7, s^8, s^9\}$. Then, the points in L whose projections p^* dominates (s_8) are removed from L, and $s^{12}.cy$ is set to $\mathsf{next}_y(s^{12}.cx, \mathrm{L}) = s^7$.

Hypervolume Updates

Having established how to update the HV3D$^+$ data structure in linear time, it is clear that UpdateHypervolume can also be computed in linear time by using Algorithm 4.3 to recompute Hypervolume after updating the data structure. Alternatively, the value of the hypervolume indicator can be updated with the contribution of the point u to be added to, or removed from, Q. Although the contribution of u can be computed while the data structure is being updated in both cases, for simplicity only the case where the point has not yet been added to or has already been removed from Q is explained here.

The OneContribution computation begins with the construction of a list containing all points in $\{p \in \mathrm{Q} \mid p_z \leq u_z\}$ whose projections are inner or outer delimiters of the contribution of u^* to the projection of that set. This list, L, can be set up in linear time as in Algorithm 4.3, by sweeping Q while $p_z \leq u_z$. Having constructed L, the contribution of u^* to L* is computed and stored in a variable, *area*, and points $p \in \mathrm{Q}$ such that $p_z > u_z$ are visited in ascending order of p_z until a point such that $p^* \leq u^*$ is found.

For each visited point, p, the contribution of u is updated by accumulating the product of *area* by the height of the current slice in another variable, *vol*, and *area* is updated by subtracting the joint contribution $H(p^*, u^*, \mathrm{L}^*)$ from it. $H(p^*, u^*, \mathrm{L}^*)$ is computed by calling either $\mathsf{computeArea}_y(p \vee u, \mathrm{L}, p.cx)$ or $\mathsf{computeArea}_x(p \vee u, \mathrm{L}, p.cy)$, depending on the relative position of p with respect to u. Then, all points that are no longer outer or inner delimiters of the contribution of u^* on the plane $z = p_z$ are removed from L, and p is added to L if it made *area* decrease. When u does not dominate p, computing the joint contribution of p^* and u^* and removing the required points from L always entails starting at one end of L, which requires linear time in the number of points removed. On the other hand, when $u \leq p$, such a linear time operation is made possible by the availability of attributes $p.cx$ and $p.cy$, as before. Therefore, the time complexity of the whole OneContribution computation amortizes to $O(n)$.

The procedures used to compute a three-dimensional contribution in HV3D$^+$ and in HV4D are rather alike, and although the latter is tightly integrated in the main algorithm, it also considers both the incremental and the decremental scenarios, and could easily be made available standalone. The main differences lie in the data structures used (a list of points versus a list of boxes) and, more crucially, in how dominated points are handled. In HV4D, computing the three-dimensional contribution of a point u to the current set of points requires removing any points dominated by u from the current set first and computing their contributions, as well. Although this is done in linear time per point when there are no dominated points, and amortizes to linear time per point over a complete four-dimensional Hypervolume computation, the complexity of computing a single contribution in HV4D is not linear in general, whereas it is always linear in HV3D$^+$.

Another advantage of HV3D$^+$ is that it can be directly used to recompute the hypervolume indicator with Algorithm 4.3 after moving the reference point, as long as it remains strongly dominated by every point stored in the data structure.

4.1.2 Four Dimensions

The hypervolume indicator in four dimensions can be computed in $O(n^2)$ time by performing a sequence of HV3D$^+$ updates, leading to a new algorithm, HV4D$^+$. As in HV4D, points in $\mathrm{X} \subset \mathbb{R}^4$ are visited in ascending w-coordinate order, dividing the dominated region into n slices. For each visited point, p, the volume of the three-dimensional base of the current slice is computed by adding the contribution of p^* to the volume of the previous slice. Because p^* may dominate points in X^* visited previously, handling dominated points is a must. Not having to remove points whose projections are dominated is an important feature in extending the approach to the computation of all contributions in four dimensions (see Section 4.2.2).

4.1.3 Experimental Results

To evaluate the potential impact of the proposed algorithms in practice, they were evaluated experimentally and compared to their most direct competitors in the literature, considering a number of relevant scenarios and concrete data sets.

Experiment Setup

The proposed algorithms were implemented in C. All codes used in the experiments [1] were compiled with gcc 5.3.1 and flags -march=corei7 -O3. Tests were run on an Intel Core i7-3612QM 2.10GHz CPU with 6 MB cache and 8 GB of RAM.

To evaluate the performance of the algorithms, *cliff* and (concave) *spherical* data sets (see Section 3.5) containing 10^5 points each were generated at random. All smaller sets of points were generated by sampling the initial sets of 10^5 points at random. Additionally, *hard* data sets for $d = 4$ (see Section 3.5) were generated for every set size $n \leq 10^5$ considered. The reference point used was $(1, \ldots, 1)$.

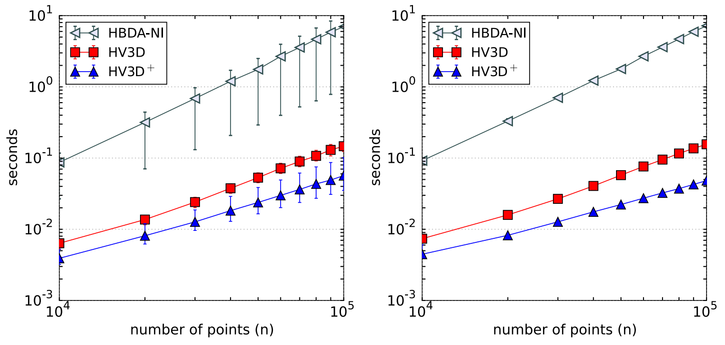
The plots presented next show runtimes for growing numbers of points on the above types of data sets. Because many algorithms for the hypervolume indicator are sensitive to objective reordering, each data point and the corresponding error bar on a plot represent the average, minimum and maximum runtimes over all permutations of the objectives for a single set instance (6 permutations for $d = 3$, and 24 for $d = 4$). Due to the computational effort required, smaller sets of up to 10^4 points were considered in some experiments. In general, greater variability was observed on cliff data sets than on spherical data sets.

Runtimes

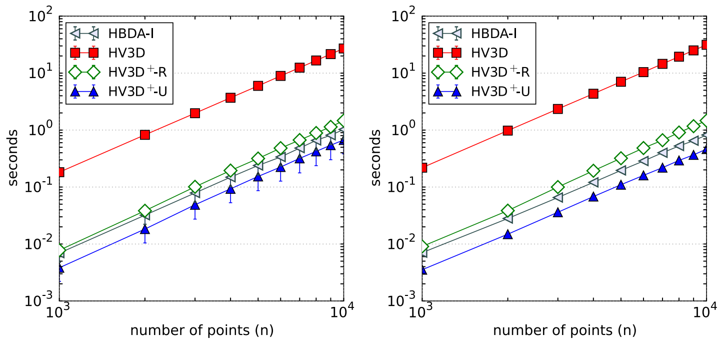
Figure 4.3 (a) shows that HV3D$^+$ is generally faster than the original HV3D at computing Hypervolume in three dimensions, despite sweeping input sets twice. The runtime of HBDA-NI appears to grow quadratically, as expected.

Results for the case of an initially empty unbounded archive whose hypervolume indicator value was updated each time a single point from a given test set was added to it are presented in Figure 4.3 (b). Since all test sets contained only nondominated points, the size of the archive increased with every new point. Concerning HV3D, the hypervolume indicator was computed n times for growing archive size $k = 1, \ldots, n$,

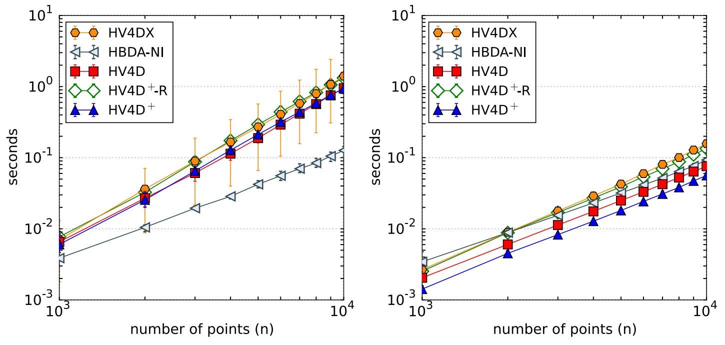
[1]The source code for the proposed HV3D$^+$ and HV4D$^+$ algorithms is available within the HVC package at `https://github.com/apguerreiro/HVC`.



(a) HYPERVOLUME computation for $d = 3$



(b) Sequential incremental UPDATEHYPERVOLUME for $d = 3$



(c) HYPERVOLUME computation for $d = 4$

Figure 4.3: Runtime performance of algorithms on different hypervolume indicator problems and data sets: cliff (left) and spherical (right).

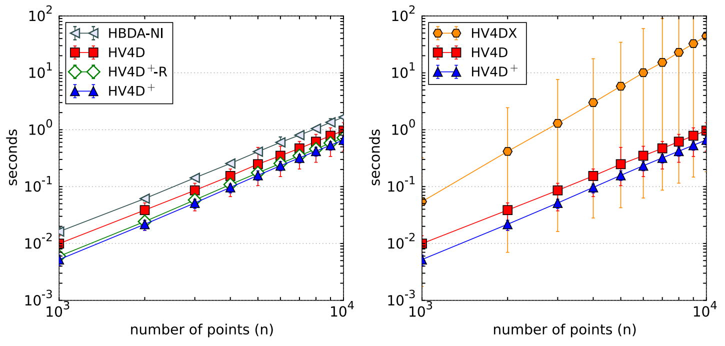


Figure 4.4: Runtime performance of algorithms for HYPERVOLUME computation for $d = 4$ on the hard data set.

resulting in an $O(n^2 \log n)$-time algorithm for this scenario. This is compared to HV3D$^+$ linear-time updates, where recomputing the hypervolume indicator for each new point is denoted by HV3D$^+$-R, and computing and adding the contribution of the new point to the current value of the indicator is denoted by HV3D$^+$-U. Both update approaches take $O(n^2)$ time on the whole problem. It can be seen that both HV3D$^+$-R and HV3D$^+$-U clearly outperform HV3D, with speed-ups of up to 23 and 47 times on cliff data sets, and up to 25 and 68 times on spherical data sets, respectively. HV3D$^+$-U was up to 3 times faster than HV3D$^+$-R. HBDA-I is slightly faster than HV3D$^+$-R, but is still outperformed by HV3D$^+$-U, which was up to twice as fast as HBDA-I.

Finally, Figures 4.3 (c) and 4.4 show that the default HV4D$^+$ (based on single contribution updates) and a variant HV4D$^+$-R based on full recalculation updates are competitive with HV4D for Hypervolume computation in four dimensions, with HV4D$^+$ generally matching or exceeding the performance of HV4D. Although the quadratic time complexity of HBDA-NI for $d = 4$ can be observed on the hard data set (Figure 4.4, left), it exhibited sub-quadratic behavior on the cliff data set, where it clearly outperformed the other algorithms. Nevertheless, it was up to 3 times slower than HV4D$^+$ on the other data sets.

Finally, the claimed [50] performance improvement of HV4DX (see [A12]) over the $O(n^2)$-time HV4D algorithm[2] could not be observed. Not only was it up to twice as slow on average as the original HV4D implementation on the cliff and spherical data sets (Figure 4.3 (c)), it also exhibited cubic runtime growth on the hard data set (Figure 4.4, right).

4.1.4 Practical Implications

The HV3D$^+$ algorithm matches the current upper bound of $O(n)$ on the time complexity for computing the UpdateHypervolume problem in the $d = 3$ case if $S \cup \{p\}$ is a nondominated point set (a time complexity obtained with UHV3D [B5]). On the other hand, if S contains points dominated by p (in the incremental case), then HV3D$^+$ improves the current upper bound from $O(n \log n)$ to $O(n)$. Moreover, HV3D$^+$ also improves the time-complexity of the problem of recomputing the hypervolume indicator after changing the reference point from $O(n \log n)$ to $O(n)$. Practical implications of these algorithms include at least the following:

- Potentially fast/faster implementation of benchmarking tools such as BBOB [42], with efficient on-the-fly updates.
- Faster (re)computation of the hypervolume indicator after changing the reference point.
- Extendable to the analogous all-contributions cases (see next Section).

In addition, the proposed data structures, preprocessing and computation techniques are likely to be useful in the development of more efficient algorithms for other related problems, namely for the computation of local upper bounds (see Section 3.4.1), and for the computation and update of the Expected Hypervolume Improvement (EHVI) [132].

[2]Version 1.2 [80] was used.

4.2 Computation of All Hypervolume Contributions

HV3D$^+$ lends itself to further extension to the computation and update of all hypervolume contributions. A new AllContributions algorithm for three dimensions, named HVC3D, supports contribution updates in linear time, and is proposed next. Two update scenarios are considered:

Nondominated sets Any dominated points are ignored and/or removed as described in Subsection 4.1.1 (in "Preprocessing" and "Data Structure Updates") and do not affect the value of the individual contributions of nondominated points. This is sufficient to implement SMS-EMOA, for example.

Dominated points A new point may dominate existing ones (incremental scenario), in which case its individual contribution is also delimited by the points it dominates. This allows a new $O(n^2)$ algorithm to be constructed for the AllContributions problem in four dimensions.

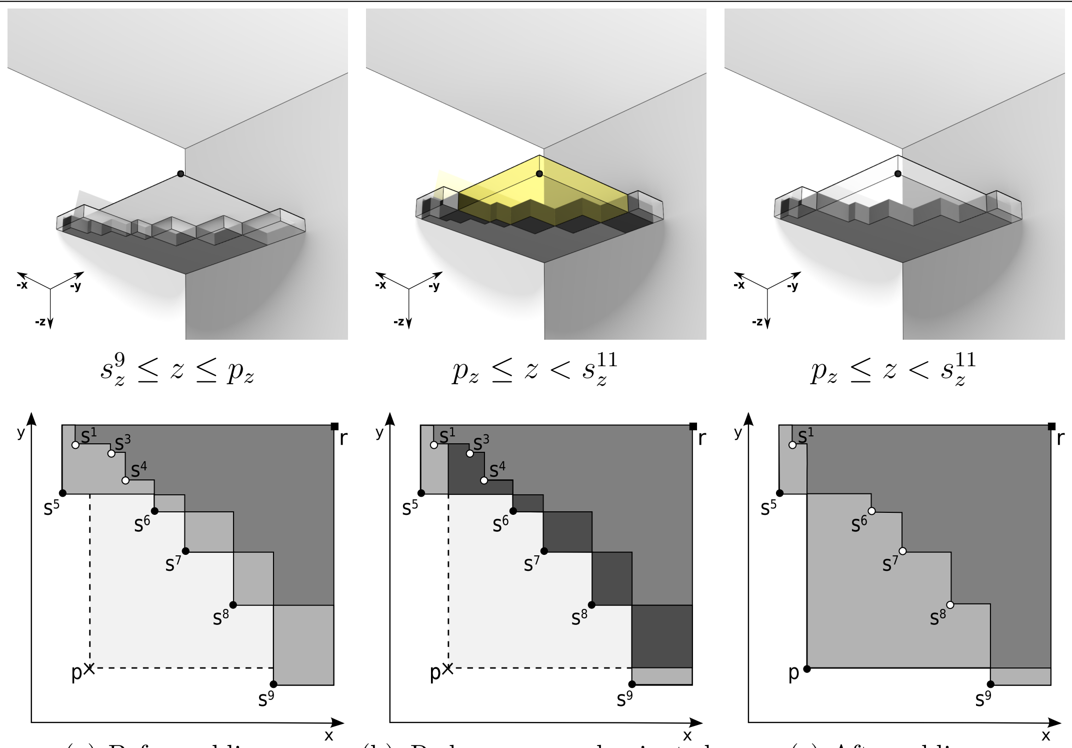
More general scenarios involving dominated points can in principle be addressed using similar techniques, but their relevance is not clear at present.

4.2.1 Three Dimensions

Data Structures

To support the computation of individual contributions, the HV3D$^+$ data structure is extended with additional point attributes. For each point $p \in \mathrm{Q}$, the area of the current base of its contribution is stored in $p.a$, the current volume of this contribution is stored in $p.v$, and the z-coordinate value up to which the volume $p.v$ has been computed is stored in $p.z$. Moreover, to support the handling of dominated points, the number of points that dominate p and a pointer to one of those points are stored in $p.nd$ and $p.dom$, respectively. Since the initial set of points, X, is required to contain only nondominated points, $p.nd$ is initialized to zero and $p.dom$ is set to NULL. Whenever a point that dominates p is added to Q, $p.nd$ is incremented, and $p.dom$ is set to that point. As discussed later in Subsection 4.1.4, points which become dominated by more than one point will be discarded.

In HVC3D, the set of points visited so far whose projections on the (x, y)-plane are nondominated is maintained in a doubly-linked list, L, sorted in ascending y-coordinate order (and, therefore, also in descending order of the x coordinate). This list plays exactly the same role as L in HV3D$^+$. Moreover, each point $q \in \mathrm{L}$ maintains in q.L a list of the points visited so far whose projections are outer or inner delimiters of the current exclusive contribution of q^*. In the example of Figure 4.5 (a), before p is processed, L contains $s^5, \ldots, s^9$, i.e., the points delimiting the base (bottom figure) of the last slice processed (top figure). The current exclusive contribution of each of these points in L to that slice are depicted as transparent volumes in the top figure, and their bases are depicted in medium gray in the bottom figure, facilitating the identification of each one's outer and inner delimiters. In particular, list s^5.L contains points s^6, s^4, s^3, s^1, and a sentinel, in this order. These are the delimiters of the base of the contribution of s^5 in that slice as can be observed in Figure 4.5 (a).



(a) Before adding p (b) Reduce areas dominated by p^* (c) After adding p

Figure 4.5: Contribution of each point, before adding a point p, when the contributions are decreased, and after adding p.

Finally, a new operation based on Algorithm 4.1 is defined:

updateVolume$_y(p, \mathrm{S}, s)$ If $s = \mathsf{outerDelimiter}_x(p, \mathrm{S})$, for each point $q \in \mathrm{S}$ whose projection q^* is an outer or inner delimiter of the contribution of p^* to S^*, add the volume of the current contribution slice, $q.a \cdot (p_z - q.z)$, to $q.v$, and set $q.z = p_z$.

In this case, procedureA, procedureB and procedureC in Algorithm 4.1 all perform the same operations, but on different points, respectively $s = \mathsf{outerDelimiter}_x(p, \mathrm{S})$, all $q \in \mathrm{S}$ such that $p^* \leq q^*$, and $e = \mathsf{outerDelimiter}_y(p, \mathrm{S})$. Just like the other operations based on Algorithm 4.1, updateVolume requires $O(t)$ time, where t is the number of points dominated by p^* in S^*. In the example of Figure 4.5 (a), given p, $\mathrm{L} = \{s^5, \ldots, s^9\}$ and $s = s^9$, updateVolume(p, L, s) updates the values of attributes v and z of points $s^5, \ldots, s^9$.

Computing All Contributions

As an extension of HV3D$^+$ to the computation of AllContributions, HVC3D consists of an $O(n \log n)$-time preprocessing step, identical to Algorithm 4.2, followed by an actual computation step, which is detailed in Algorithm 4.4. Points $p \in \mathrm{Q}$ are visited in ascending z-coordinate order, as in Algorithm 4.3. Considering that all points are nondominated (first scenario), each point is processed in lines 6 to 21 of Algorithm 4.4.

The processing of nondominated points is divided into three main parts. In the first part (lines 6 to 11), the volumes associated with the outer and inner delimiters of p^* in L^* are updated, and the base area of the contribution of p, depicted in

Algorithm 4.4 HVC3D – AllContributions Computation

```
Require: $\mathrm{X} \subset \mathbb{R}^3$ // a set of $n$ points
Require: $r \in \mathbb{R}^3$ // the reference point
Require: Q // a linked list sorted in ascending order of dimension $z$ containing $\mathrm{X} \cup \{(-\mathrm{REALMAX}, -\mathrm{REALMAX}, r_z)\}$ with $p.cx$, $p.cy$, $p.nd$ and $p.dom$ set for all $p \in \mathrm{Q}$
 1: $\mathrm{L} \leftarrow \{(r_x, -\infty, -\infty), (-\infty, r_y, -\infty)\}$ // linked list sorted in ascending order of dimension $y$
 2: for each $p \in \mathrm{Q}$ do
 3:   $p.v \leftarrow 0$
 4:   $p.z \leftarrow p_z$
 5:   if $p.nd = 0$ then
 6:     $\mathsf{updateVolume}_y(p, \mathrm{L}, p.cx)$
 7:     $p.a \leftarrow \mathsf{computeArea}_y(p, \mathrm{L}, p.cx)$
 8:     $p.\mathrm{L} \leftarrow \mathsf{removeDominated}_y(p, \mathrm{L}, p.cx)$
 9:     $\mathsf{add}_y(p.cx, p.\mathrm{L}, \mathrm{NULL})$
10:     $\mathsf{add}_x(p.cy, p.\mathrm{L}, \mathrm{NULL})$
11:     $\mathsf{add}_y(p, \mathrm{L}, p.cx)$
12:     $q \leftarrow p.cy$
13:     $q.a \leftarrow q.a - \mathsf{computeArea}_y(p \vee q, q.\mathrm{L}, \mathsf{head}_y(q.\mathrm{L}))$
14:     $\mathsf{removeDominated}_y(p \vee q, q.\mathrm{L}, \mathsf{head}_y(q.\mathrm{L}))$
15:     $\mathsf{remove}(\mathsf{head}_y(q.\mathrm{L}), q.\mathrm{L})$
16:     $\mathsf{add}_y(p, q.\mathrm{L}, \mathrm{NULL})$
17:     $q \leftarrow p.cx$
18:     $q.a \leftarrow q.a - \mathsf{computeArea}_x(p \vee q, q.\mathrm{L}, \mathsf{head}_x(q.\mathrm{L}))$
19:     $\mathsf{removeDominated}_x(p \vee q, q.\mathrm{L}, \mathsf{head}_x(q.\mathrm{L}))$
20:     $\mathsf{remove}(\mathsf{head}_x(q.\mathrm{L}), q.\mathrm{L})$
21:     $\mathsf{add}_x(p, q.\mathrm{L}, \mathrm{NULL})$
22:   if $p.nd = 1$ then
23:     $q \leftarrow p.dom$
24:     $q.v \leftarrow q.v + q.a \cdot (p_z - q.z)$
25:     $q.z \leftarrow p_z$
26:     $q.a \leftarrow q.a - \mathsf{computeArea}_y(p, q.\mathrm{L}, p.cx)$
27:     $\mathsf{removeDominated}_y(p, q.\mathrm{L}, p.cx)$
28:     $\mathsf{add}_y(p, q.\mathrm{L}, p.cx)$
```

light gray in Figure 4.5 (a), is computed. Then, the points whose projections are dominated by p^* (s^8, s^7, s^6) are moved from L to p.L, as their contribution is zero above the plane $z = p_z$, and copies of $p.cx$ and $p.cy$ (s^9 and s^5), corresponding to the outer delimiters of p^*, are added at each end of p.L, so that the contribution of p can be updated efficiently in subsequent iterations. Finally, p is inserted into L.

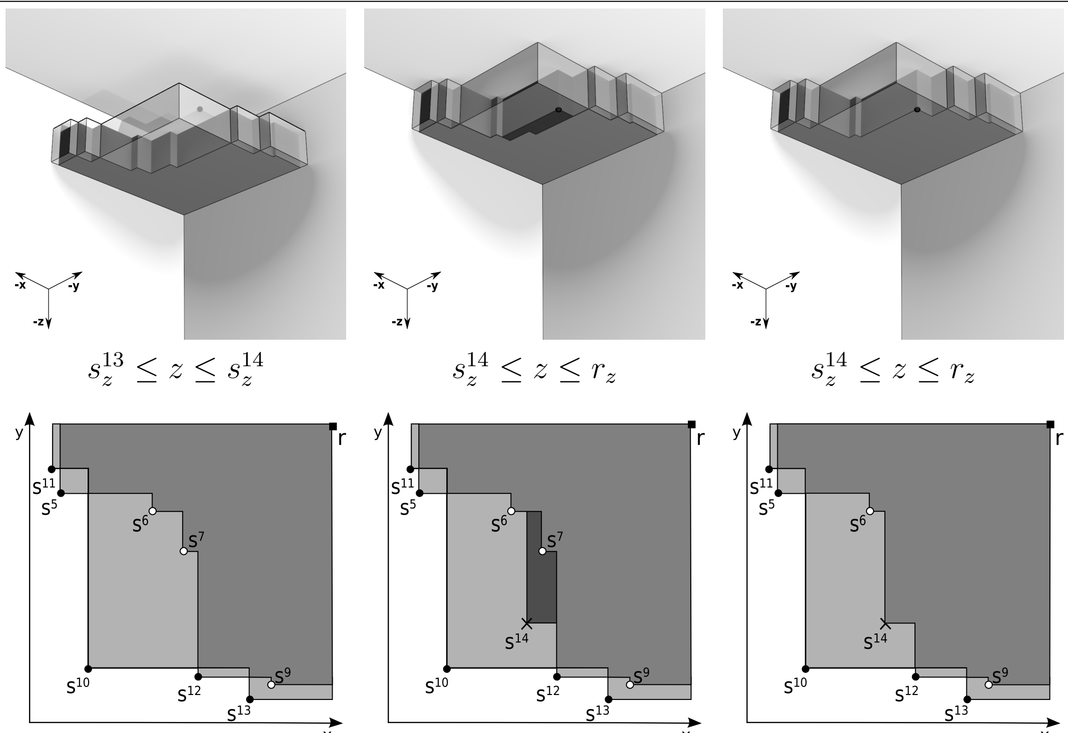
In the second part (lines 12 to 16), the base area of the contribution of point $q = p.cy$ is updated, as p^* dominates part of the region dominated by q^*. To that end, the joint contribution of q^* and p^* to q.L* is subtracted from $q.a$, the points whose projections are dominated by p^* are removed from q.L, and the head of this list is replaced by p. Now referring to Figure 4.5 (b), $q = s^5$ and q.L contains $\{s^6, s^4, s^3, s^1\}$ plus the sentinel. The joint contribution of p and q to q.L in the next slice (between p_z and s^{11}_z) is depicted as the dark region dominated by both points and whose base is depicted in the bottom figure. Its computation (line 13) involves visiting the points from s^6 to s^1. Then, s^4 and s^3 are discarded (line 14) and s^6 is replaced by p at the head of q.L (lines 15 and 16). At this point, q.L contains the outer and inner delimiters of the exclusive contribution of q^* at $z = p_z$, namely points p, s^1 and the sentinel, as Figure 4.5 (c) illustrates.

The base area of the contribution of $p.cx$ is updated analogously in the third part (lines 17 to 21). The last point in Q to be visited is $(-\text{REALMAX}, -\text{REALMAX}, r_z)$, the sentinel, which forces the computation of the contributions of all input points to complete and the results for all points $p \in \text{X} \subset \text{Q}$ to be made available in $p.v$ at the end of the run. Here, $-$REALMAX denotes a finite value less than any coordinate of any input point.

Note that the contribution of a point is updated only when its contribution to the (x, y)-plane decreases, which happens only if it is an outer or inner delimiter of the current point p. In the volume update in line 6 when, for example, the outer delimiter $q = s^5$ of p is updated, $q.a$ stores the area of the region in medium gray dominated by s^5 in the bottom Figure 4.5 (a) and $q.z$ contains the z-coordinate value at which $p.a$ was last updated (and up to which $q.v$ is also updated), in this case, $q.z = s^6_z$. Thus, the value $q.a \cdot (p_z - q.z)$ added to $q.v$ in updateVolume corresponds to the contribution of s^5 to all slices between s^6_z and s^{10}_z, including the contribution depicted at the top of Figure 4.5 (a). The value $q.a$ is updated in line 13 to correspond to the base of its contribution in the next slice (see Figure 4.5(c)) and $q.z$ is set to p_z. As the outer delimiters of p have contribution above p_z, they are kept in L while its inner delimiters are removed from L as their contributions are fully computed at $z = p_z$. All of these inner and outer delimiters are still needed to compute the contribution of p above $z = p_z$ and so they are kept in p.L.

In the second scenario, Q may contain dominated points due to incremental updates to the data structure. Such updates are performed exactly as described in Subsection 4.1 with respect to point attributes cx and cy, but any points $p \in \text{Q}$ which are dominated by a new point u to be inserted are also marked as such by incrementing $p.nd$ and setting $p.dom = u$. Note that, although the hypervolume contribution of a dominated point is zero by definition, if it is dominated by a single point, it also decreases the contribution of that point, which is why such dominated points must remain in Q. On the other hand, points dominated by two or more points do not affect the exclusive contribution of any of those points, and can be discarded. The insertion of new dominated points is not considered in this scenario.

Points $p \in \text{Q}$ which are dominated by a single point ($p.nd = 1$) are processed


$s_z^{13} \leq z \leq s_z^{14}$
$s_z^{14} \leq z \leq r_z$
$s_z^{14} \leq z \leq r_z$
-x -y -z
x
y
r
s^{11}
s^{5}
s^{6}
s^{7}
s^{14}
s^{10}
s^{12}
s^{9}
s^{13}

(a) Before adding s_z^{14} (b) Reduce areas dominated by s_z^{14*} (c) After adding s_z^{14}

Figure 4.6: Contribution of each point, before adding the dominated point s^{14}, when the contribution of s^{10} is decreased, and after adding s^{14}.

in lines 23 to 28. After updating the volume $q.v$ associated with $q = p.dom$, the contribution of p^* to $q.\mathrm{L}^*$ is computed and subtracted from $q.a$. Then, any points in $q.\mathrm{L}$ whose projections are simultaneously dominated by p^* and q^* are discarded, and p is inserted into $q.\mathrm{L}$. In the example of Figure 4.6,[3] $p = s^{14}$ is dominated (only) by s^{10} and the contribution that is not retained by s^{10} because of s^{14} is represented in dark gray in Figure 4.6(b). Therefore, $p.nd = 1$ and $q = p.dom = s^{10}$. Also, $p.cx = s^{12}$, $p.cy = s^6$ and $q.\mathrm{L} = \{s^{12}, s^7, s^6, s^5\}$. The contribution of p^* to $q.\mathrm{L}^*$ (line 26) is computed by visiting points s^{12}, s^7 and s^6. Then, s^7 is discarded (line 27) and p is added to $q.\mathrm{L}$ (line 28). At the end of the iteration, $s^{10}.\mathrm{L} = \{s^{12}, s^{14}, s^6, s^5\}$ contains the points whose projections are delimiters of s^{10*} at $z = s_z^{14}$.

In Algorithm 4.4, each point in $p \in \mathrm{Q}$ is added to L (line 11) once. Moreover, each p other than the sentinel is added at most once to a list associated with a point in L as an inner delimiter (lines 8 or 28). Although the same point may be an outer delimiter (cx or cy) of several other points in Q, at most four outer delimiters are added to lists associated with points in L in each iteration ($p.cx$ and $p.cy$ are added to $p.\mathrm{L}$ in lines 9 and 10, and p is added to two lists in lines 16 and 21). Therefore, at most $(n+1)+n+4(n+1) = 6n+5$ points are added to lists, which is also an upper bound on the number of points removed. In each iteration of the algorithm, points other than p are visited only to update the corresponding volumes or to update some area. This involves $O(1)$-time operations on each of those points through calls to updateVolume and computeArea. Every point visited in those calls is either discarded right after (in removeDominated) or is an outer delimiter, and the number of outer delimiters visited in each call is a constant. Consequently, the time-complexity of

[3]Note that point s^9 was missing in Figure 4(d) in [83].

Algorithm 4.4 amortizes to $O(n)$.

Updating All Contributions

By updating the HVC3D data structure and recomputing all contributions with Algorithm 4.4, the UpdateAllContributions problem can now be solved in $O(n)$ time. Moreover, analogously to HV3D$^+$, HVC3D can be directly used to update all contributions in linear time after the reference point is changed. Although HVC3D extends the HV3D$^+$ data structure with additional point attributes, attributes *cx* and *cy* are still updated in linear time exactly as described in Subsection 4.1. When new points may dominate existing ones, attributes *nd* and *dom* also need to be updated as explained in the previous subsection, which is also carried out in linear time. All other point attributes are for Algorithm 4.4's own use, and do not need to be updated on single point insertion or removal.

A potentially faster alternative is suggested by the very definition of Problem 3.7 (UpdateAllContributions). It consists in computing the contribution of the point u to be added to Q as described in Subsection 4.1 (which is not needed when removing a point), as well as the joint contributions of u and p to Q, for each *strong* delimiter $p \in$ Q of the contribution of u, thus avoiding unnecessary computations. Going back to the example depicted in Figure 4.1 (b), if the contributions of the points in X $= \{s^1, \ldots, s^9, s^{11}, \ldots, s^{14}\}$ are known and $u = p$ is the new point to be added to X, the contributions of $s^1, \ldots, s^4, s^{11}$ and s^{13} remain unchanged, and there is no need to recompute them.

Since it may be difficult to know in advance which delimiters are strong and whether an outer delimiter is a proper delimiter or not, let D^1 denote the set of all inner and outer delimiters of u in Q. Points in D^1 are used to compute the contribution of u, but they are also the points whose exclusive contribution may be decreased by the addition of u to Q. Therefore, in order to update their joint contribution with u, their own inner and outer delimiters must be considered. The set of points that are inner and outer delimiters of the joint contribution between u and individual points in D^1, but not of the contribution of u, will be denoted by D$^2 \subseteq$ Q $\setminus$ D^1. The remaining points in X $\setminus$ (D$^1 \cup$ D^2) can be ignored. In Figures 4.1(a) and 4.1(b), D$^1 = \{s^5, \ldots, s^9, s^{12}, s^{14}\}$, D$^2 = \{s^1, s^3, s^4, s^{11}, s^{13}\}$, and s^2 can be ignored. Note that points s^1, s^3, s^4 and s^{11} are in D^2 because they delimit the contribution of $(u \vee s^5)$, but not that of u.

Recall the computation of the contribution of u to Q as described in Subsection 4.1. After saving the current contributions and setting $p.v = 0$ for all $p \in$ Q, the points in $\{p \in \mathrm{Q} \mid p_z \leq u_z\}$ whose projections are inner or outer delimiters of the contribution of u^* to the projection of that set are added to list L $\subseteq$ D^1. Then, for each $q \in$ L, a list q.L is constructed containing the outer and inner delimiters of the joint contribution between u^* and q^* at $z = u_z$. All of these lists can be set up in linear time as in Algorithm 4.4, by sweeping Q while $p_z \leq u_z$. In Figure 4.5 (a), considering $u = p$, L $= \{s^5, \ldots, s^9\}$. Then, for each point $q^* \in$ L*, the area of the region previously dominated exclusively by q^* that is also dominated by u^* at $z = u_z$ is computed in linear time, and stored in $q.a$. These regions are depicted in dark gray in Figure 4.5 (b).

After these initialization steps, the points $p \in$ Q such that $p_z > u_z$ are visited in ascending order of p_z. These points are skipped unless they decrease the area associated with u or with any of the points in L. The joint contribution between p^*

and u^* is computed and stored in $p.a$. Moreover, for each point $q \in \mathrm{L}$, the associated volume is updated as usual, and the area of the region jointly dominated by q^*, u^* and p^* is computed and subtracted from $q.a$.

The algorithm ends once all points in Q have been visited or after two points dominating u^* are encountered, as from that point onwards the joint contribution of u and any other point must be zero. Finally, if u is to be added to Q, the computed joint contributions are subtracted from the corresponding original values, and the HVC3D data structure is updated. If u was removed from Q, then the data structure has already been updated, and it suffices to add the computed joint contributions to their corresponding previous values.

4.2.2 Four Dimensions

Similarly to HV4D$^+$, a new algorithm named HVC4D for the AllContributions problem in four dimensions is obtained by performing a sequence of HVC3D updates, as follows. Given a nondominated point set $\mathrm{X} \subset \mathbb{R}^4$, points in $\mathrm{X} \cup \{(-\infty, -\infty, -\infty, r_w)\}$ are sorted in ascending w-coordinate order, stored in a linked list Q, and visited in that order. For each visited point, p, the contribution of p^* to the projection, S^*, of the (initially empty) set, S, of points visited before p is computed, and the contributions of its inner and outer delimiters in $\mathrm{S}^* \cup \{p^*\}$ are updated as described in Subsection 4.1.4. Then, p is added to S, and the contribution of each point in S^* is multiplied by the difference between the w-coordinate of the next point in Q and p_w to obtain the hypervolume of the current four-dimensional slice of each individual contribution. Slice hypervolumes are accumulated separately for each point to obtain the corresponding contributions in four dimensions. Since the algorithm performs n linear-time HVC3D updates, the time complexity of this algorithm is $O(n^2)$.

Handling points which are dominated in three dimensions is required to correctly compute AllContributions in four dimensions because a point $q^* \in \mathrm{S}^*$ which is dominated by a single point $p^* \in \mathrm{S}^*$ decreases the contribution of p^*. Since dominated points are not considered in the EF algorithm, iterating over EF without modification to compute all contributions in four dimensions would not work.

4.2.3 Experimental Results

As in Section 4.1.3, the proposed algorithms[4] were compared to the most direct competitors with available implementations, and the same data sets were considered, namely, cliff, (concave) spherical and hard data sets. All algorithms were implemented in C and the experimental setup was identical to that of Section 4.1.3.

Runtimes

Regarding the computation of AllContributions in three dimensions, it can be observed in Figure 4.7 (a) that HVC3D remains competitive with EF, as expected. HVC3D and EF were 2 to 14 times faster than exQHV. In comparison to a dedicated

[4]The source code for the proposed HVC3D$^+$ and HVC4D$^+$ algorithms is available within the HVC package at `https://github.com/apguerreiro/HVC`.

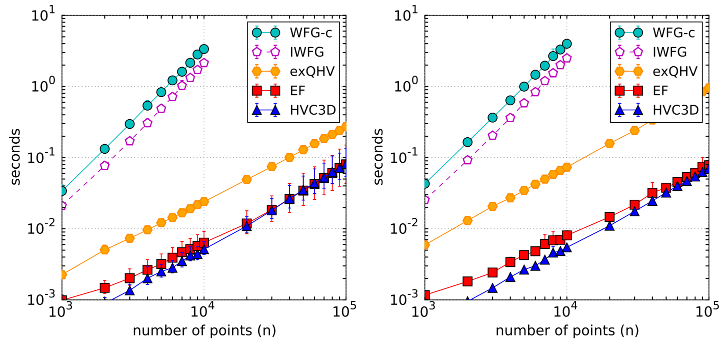
adaptation of WFG[5], here referred to as WFG-c, they were 34 to 728 times faster in the tests performed. Since computing AllContributions can also be used to identify a least contributor, IWFG was included in the comparison, but it was nevertheless 21 to 456 times slower than HVC3D or EF.

Figure 4.7 (b) shows the runtimes for the sequential update of all contributions in three dimensions on the unbounded archive setup described earlier in connection with Figure 4.3 (b). Similarly to HV3D in that case, EF was called n times when filling an archive of size n, corresponding to an overall $O(n^2 \log n)$ time complexity. The two HVC3D variants, -R (recomputing) and -U (contribution updates), take overall $O(n^2)$ time due to the linear-time updates. Both HVC3D-U and HVC3D-R clearly outperform EF in this scenario, showing speed-ups of up to 56 and 21, respectively. HVC3D-U performed up to 4 times faster than HVC3D-R.

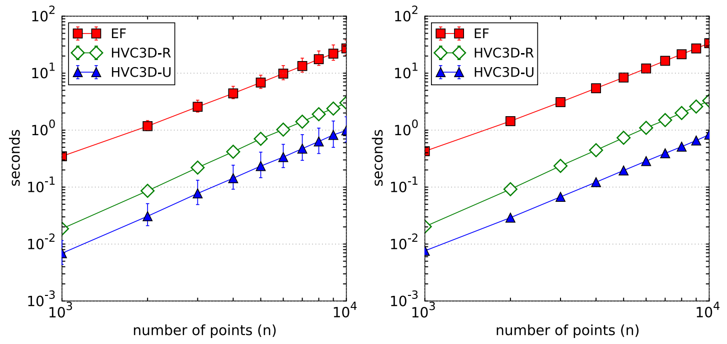
Results for the simulation of a bounded archive similar to the environmental selection process in SMS-EMOA are presented in Figure 4.7 (c). A fixed archive of size 200 was updated $n - 200$ times by each algorithm by adding a new point and then removing a least contributor. In the case of HVC3D-U, contributions were updated after adding the new point and again after removing a least contributor. On the other hand, since contributions only need to be known after a new point is added in order to identify the least contributor, with HVC3D-R only the data structure was updated when a least contributor is removed. Insertion of a new point caused a data structure update followed by the computation of all contributions. Similarly, contributions were recomputed only on point insertions when using EF. The runtimes include the computation of the initial archive with 200 solutions, in $O(n \log n)$ time in all cases. HVC3D-U and HVC3D-R were up to 50 and 30 times faster than EF, respectively. The results show that, even though contributions were recomputed by HVC3D-R only half the time, it was still up to 2 times slower than HVC3D-U.

Finally, runtimes for AllContributions in four dimensions are shown in Figure 4.8, where HVC4D is compared to exQHV, WFG-c, HV4D and HBDA-NI. HV4D and HBDA-NI are called $n + 1$ times for each set, and therefore the whole computation has a time complexity of $O(n^3)$. As expected, HVC4D significantly outperformed both HV4D and HBDA-NI, with observed speed-ups ranging between 45 and 1069 and between 81 and 2270, respectively. HV4D and HBDA-NI were also significantly outperformed by WFG-c, but HVC4D was still 3 to 372 times faster than WFG-c. IWFG was also included for reference. Note that computing all contributions with WFG-c took fairly the same time as identifying the least contributor with IWFG. The exQHV algorithm seems the most sensitive to the data. It showed almost linear behavior in the cliff and, in particular, the spherical data set, but also showed slightly worst than quadratic behavior in the hard data set. Although exQHV has a tendency to outperform HVC4D in the large cliff and spherical data sets, it was, in general, slower than HVC4D in the tests performed. In particular, exQHV was able to perform twice as well as HVC4D with 10^4 in the cliff data set, but otherwise was up to 10 times slower on the cliff and spherical data sets. On the hard data set, HVC4D was 21 to 93 times faster than exQHV.

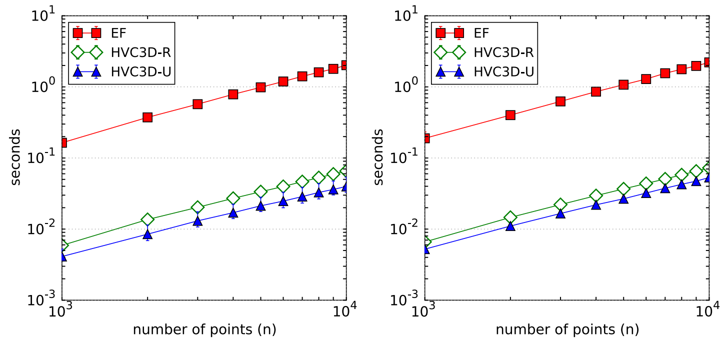
[5]Version 1.11 of WFG was adapted to iterate over the function used to compute the contribution of a single point, which is faster than iterating over WFG as such.



(a) AllContributions computation for $d = 3$



(b) Sequential incremental UpdateAllContributions for $d = 3$



(c) Sequential UpdateAllContributions for $d = 3$

Figure 4.7: Runtime performance of algorithms on different all contribution problems and data sets: cliff (left) and spherical (right).

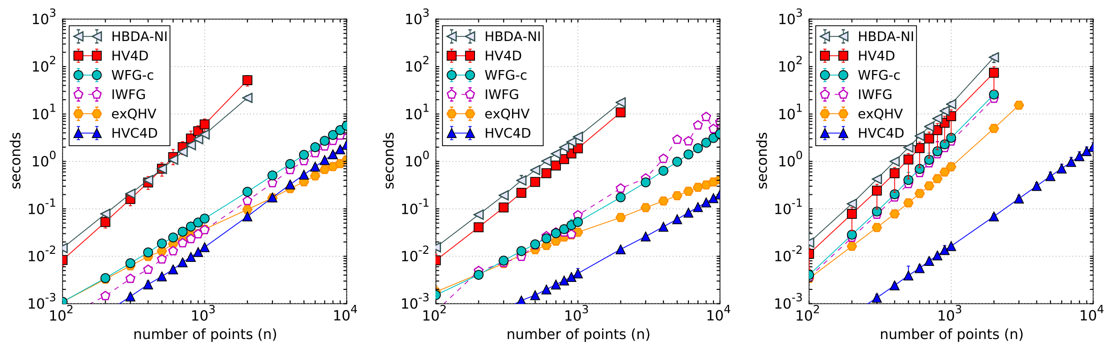


Figure 4.8: Runtime performance of algorithms for all contributions in four dimensions on the cliff (left), spherical (middle) and hard (right) data sets.

4.2.4 Practical Implications

The HVC3D algorithm improves the current upper bound on the time complexity of UpdateAllContributions in three dimensions from $O(n \log n)$ to $O(n)$. Using this algorithm in HVC4D improves the current upper bound on the complexity of AllContributions in four dimensions from $O(n^2 \log n)$ to $O(n^2)$ time. Practical implications of these algorithms include at least the following:

- Faster implementation of the decremental greedy approximation to the HSSP, and consequently, of the exact solution to the HSSP for $k = n-1$, in three and four dimensions, which can now be computed in at most $O(n(n-k) + n \log n)$ and $O(n^2(n-k))$ time, respectively.
- Faster implementation of archive-based EMO algorithms such as SMS-EMOA, also in three and four dimensions, even when the reference point is adjusted as the population moves towards the Pareto Front. Steady-state algorithms in $d = 3$ should benefit the most from the update procedures.
- Faster implementation of Branch-and-Bound techniques whose bounds rely on the solution of AllContributions problems (e.g., [75, 76]).

4.3 Decremental Greedy Algorithms for the HSSP

In this section, a general decremental greedy algorithm for the HSSP [23, 35] (see [D1]) is explained. Subsequently, specialized algorithms for 3 and 4 dimensions are proposed. In the last section, an approximation bound to the HSSP is derived.

4.3.1 General Algorithm Revisited

Given a nondominated point set $\mathrm{X} \subset \mathbb{R}^d$, where $|\mathrm{X}| = n$ and $k \in \{1, \dots, n\}$, the decremental greedy algorithm for HSSP (Problem 3.9) removes $n - k$ points from X, one at a time, always excluding the point that contributes the least hypervolume to the remaining points in X [23, 35]. Algorithm 4.5 gives the pseudo-code for such a greedy algorithm, gHSSD. In gHSSD, the contribution of each point $q \in \mathrm{X}$ to the set X itself (line 2) is computed first and stored in $q.c$. Afterwards, the point p in X with minimal contribution is selected (line 4) and is removed from X (line 5). Then, the contribution of each of the remaining points in X is updated (line 7), i.e., for each $q \in \mathrm{X}$ the portion of the hypervolume dominated by q that was dominated only by p is added to $q.c$. Lines 4 to 7 are repeated until only k points remain in X.

Note that ties may occur, i.e., at some point, more than one point may have the (same) lowest contribution. In such cases, it is correct to choose any of the tied points. However, ties that are resolved differently, will possibly result in different subsequent intermediate and final greedy solutions, both with respect to the set of points selected and, thus, to the corresponding hypervolume indicator value.

Algorithm 4.5 performs a single computation of the AllContributions problem (lines 1 and 2) and the consecutive computation of several decremental cases of UpdateAllContributions problem (lines 6 and 7). Its time complexity depends on how these computations are performed, particularly the latter. As discussed in

Algorithm 4.5 gHSSD(X, k, r)

```
Require: $\mathrm{X} \subset \mathbb{R}^d$, $k \in \mathbb{N}^+$, $r \in \mathbb{R}^d$
1: for all $q \in \mathrm{X}$ do
2:     $q.c \leftarrow H(q, \mathrm{X})$
3: for $i = 1..n - k$ do
4:     $p \leftarrow \arg\min_{q \in \mathrm{X}}\{q.c\}$
5:     $\mathrm{X} \leftarrow \mathrm{X} \backslash \{p\}$
6:     for all $q \in \mathrm{X}$ do
7:         $q.c \leftarrow q.c + H(p, q, \mathrm{X})$
8: return X
```

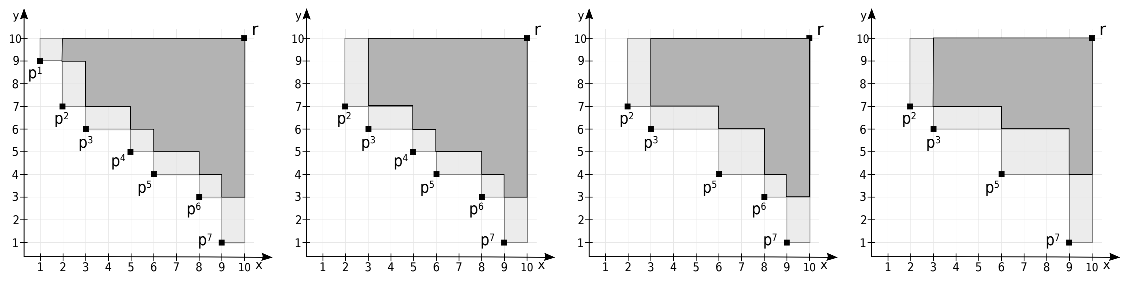
Section 3.4.5 (see description of gHSSD [D1]), using different methods for these computations lead to different instances of gHSSD. Without specific algorithms available for the UpdateAllContributions problem, the current asymptotically best instances of gHSSD in $d = 3, 4$ are obtained using state-of-the-art algorithms for the AllContributions problem. These instances of gHSSD are obtained with EF in $d = 3$ case and with BF in $d = 4$ (see Table 3.5), leading to $O((n-k)n \log n)$ and $O((n-k)n^2 \log n)$ time complexities, respectively.

4.3.2 2-Dimensional Example

Figure 4.9 shows an example of applying gHSSD to a 2-dimensional problem. Each column in the table in Figure 4.9(e) represents an iteration of the algorithm and each row shows the contribution of a point in X with respect to X itself as points are removed from X. Therefore, the cells not dashed in the i^{th} column show the points in the greedy solution of $k = n - (i-1)$, i.e., $i - 1$ points were removed, which also corresponds to an intermediate solution for $k < n - (i-1)$. A bold value in a column corresponds to the lowest contribution, indicating which point will be removed from X in that iteration. Figures 4.9(a)-(d) show the set X for the first four iterations and the contributions of the points in X.

The first step of the algorithm (line 2 in Algorithm 4.5) is to compute the contribution of every point $p \in \mathrm{X}$ to X itself, which corresponds to the first column in the table. Then, in iteration 1 of the for loop in line 3, since p^1 contributes the least to X, it is removed from X (line 5). The contributions of the points remaining in X are updated (line 7) to account for the removal of p^1 from X. For example, the contribution of p^2 is updated by adding the area of the contribution that was previously dominated exclusively by p^2 and p^1 (the area between $(2, 9)$ and $(3, 10)$). Formally, $H(p^2, \{p^2, \dots, p^7\}) = H(p^2, \{p^1, \dots, p^7\}) + H(p^2, p^1, \{p^1, \dots, p^7\}) = 2 + 1 = 3$. In this case, the contributions of $p^3, \dots, p^7$ do not change. After the contribution update step, the contributions of every point still in X are known as shown in the second column of the table. In iteration 2, p^4 is the point that contributes the least to X, and so it is removed from X and the contribution of the remaining points in X are updated. The algorithm repeats the select and update steps until k points remain in X. For a given $k < n$, the greedy solution contains the points that, in the k^{th} column from the right, correspond to cells not dashed. For example, for $k = 3$, the greedy solution can be seen in the fifth column, which corresponds to $|\mathrm{X}| = 3$, and is $\mathrm{X} = \{p^3, p^5, p^7\}$, for which $H(\mathrm{X}) = 39$.

Regarding ties, in the example of Figure 4.9, the lowest value of the x-coordinate



(a) $X = \{p^1, \ldots, p^7\}$ (b) $X = \{p^2, \ldots, p^7\}$ (c) $X = \{p^2, p^3, p^5, p^6, p^7\}$ (d) $X = \{p^2, p^3, p^5, p^7\}$

		|X|						
	$H(p, X)$	7	6	5	4	3	2	1
p	p^1	**1**	-	-	-	-	-	-
	p^2	2	3	3	**3**	-	-	-
	p^3	2	2	3	3	12	12	**28**
	p^4	1	**1**	-	-	-	-	-
	p^5	2	2	4	6	6	**8**	-
	p^6	1	1	**1**	-	-	-	-
	p^7	2	2	2	3	**3**	-	-

(e) Contributions with respect to X

Figure 4.9: Example of the decremental greedy algorithm for approximating HSSP in 2 dimensions.

was used as a tiebreaker. If a different criterion was used to untie, a different greedy solution might be obtained. For example, consider that the lowest y-coordinate was used as the tiebreaker instead. Then, the greedy solution for $k = 1$ would be $X = \{p^2\}$ and $H(X) = 24$ instead of $X = \{p^3\}$ and $H(X) = 28$, since the order of exclusion would be $p^6, p^4, p^1, p^7, p^3, p^5$ instead of $p^1, p^4, p^6, p^2, p^7, p^5$.

4.3.3 Algorithms for $d = 3, 4$

The algorithms proposed in Section 4.2 can be used to asymptotically improve instances of gHSSD for the $d = 3$ and $d = 4$ cases. In the specific case of $d = 3$, the updating feature of HVC3D (see Section 4.2.1) make it particularly suited to the gHSSD, as it can be used first to compute AllContributions and setup all the data structures and then to consecutively efficiently update all contributions (with either -R or -U versions). Therefore, by instantiating gHSSD (Algorithm 4.5) with HVC3D, the state-of-the-art time complexity is improved to $O((n-k)n + n \log n)$ time. Let us call this instance of Algorithm 4.5 by gHSSD3D. For $d = 4$, instantiating gHSSD to use HVC4D from Section 4.2.2 to compute AllContributions improves the time complexity to $O((n-k)n^2)$. Such an instance will be called gHSSD4D.

4.3.4 A Naive Bound for the Approximation Ratio to the HSSP

Although there is no approximation bound to HSSPComplement using a decremental greedy approach, that does not imply that one for the approximation of the

HSSP does not exist. In fact, an approximation bound for the HSSP, albeit a pessimistic one, can be derived. The proof is based on the observation that the contribution of the least contributor is at most $\frac{1}{|\mathrm{X}|}H(\mathrm{X})$ which can easily be shown by contradiction, noting that otherwise the sum of all contributions would be greater than $H(\mathrm{X})$. The bound is formalized as follows.

Theorem 4.1. *Given a non-empty point set* $\mathrm{X} \subset \mathbb{R}^d$ *where* $|\mathrm{X}| = n$*, a reference point* $r \in \mathbb{R}^d$ *and* $k \in \{1, \dots, n\}$*, let* S^k *and* $\mathrm{S}^{k,opt}$ *be an incremental greedy and an optimal subset of* HSSP *for size* k*, respectively. Then,*

$$\frac{H(\mathrm{S}^k)}{H(\mathrm{S}^{k,opt})} \geq \frac{k}{n} \tag{4.1}$$

Proof. Since $\mathrm{S}^n = \mathrm{X}$, and since the greedy approach always discards the point that contributes the least to the current set, then the hypervolume lost at step $t \in \{1, \dots, n-k\}$ is, at most, $\frac{1}{|\mathrm{S}^{n-t+1}|}H(\mathrm{S}^{n-t+1})$. Therefore,

$$\begin{aligned} H(\mathrm{S}^{n-t}) &\geq H(\mathrm{S}^{n-t+1}) - \frac{1}{n-t+1}H(\mathrm{S}^{n-t+1}) \\ \Leftrightarrow H(\mathrm{S}^{n-t}) &\geq \frac{n-t}{n-t+1}H(\mathrm{S}^{n-t+1}) \end{aligned}$$

Using this recurrence, the following is deduced for $k \in \{1, \dots, n-1\}$:

$$H(S^k) \geq \frac{k}{k+1}H(\mathrm{S}^{k+1}) \geq \prod_{j=k}^{n-1} \frac{j}{j+1}H(\mathrm{X}) = \frac{k}{n}H(\mathrm{X})$$

Consequently, the following proves the theorem:

$$\frac{H(\mathrm{S}^k)}{H(\mathrm{S}^{k,\mathrm{opt}})} \geq \frac{H(\mathrm{S}^k)}{H(\mathrm{X})} \geq \frac{\frac{k}{n}H(\mathrm{X})}{H(\mathrm{X})} = \frac{k}{n}$$

□

Note that, by considering X to be a nondominated point set and that every point in X strongly dominates r, then $H(\mathrm{S}^{k,\mathrm{opt}}) < H(\mathrm{X})$ holds for every $k \in \{1, \dots, n-1\}$. Thus, the bound is not tight for $k < n$.

Finally, one can conjecture of such a bound holding also for decremental greedy algorithms to approximate nondecreasing submodular functions in general. An argument similar to the one above should be enough to prove the conjecture.

4.4 Incremental Greedy Algorithms for the HSSP

In this section, a general incremental greedy algorithm for the HSSP [23], gHSS (see [D2]), is explained. Subsequently, specialized algorithms for 2 and 3 dimensions are proposed.

Algorithm 4.6 gHSS(X, k, r)

```
Require: $X \subset \mathbb{R}^d$, $k \in \mathbb{N}^+$, $r \in \mathbb{R}^d$
 1: $S \leftarrow \{\}$
 2: for all $q \in X$ do
 3:     $q.c \leftarrow H(q, S)$
 4: for $i = 1..k$ do
 5:     $p \leftarrow \arg\max_{q \in X}\{q.c\}$
 6:     $X \leftarrow X \backslash \{p\}$
 7:     for all $q \in X$ do
 8:         $q.c \leftarrow q.c - H(p, q, S)$
 9:     $S \leftarrow S \cup \{p\}$
10: return S
```

4.4.1 General Algorithm Revisited

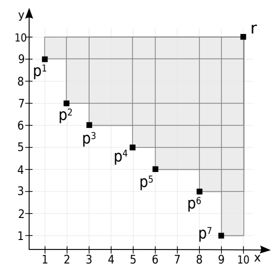
Given a nondominated point set $X \subset \mathbb{R}^d$, where $|X| = n$, in the greedy algorithm for the HSSP (Problem 3.9), $k \leq n$ points from X are chosen and stored in S, one at a time, always selecting the point that contributes the most hypervolume to the set of points already chosen. For that reason, in gHSS (Algorithm 4.6), the contribution of each point $q \in X$ to the initially empty set S (line 3) is computed first and is stored in $q.c$. Afterwards, the point p in X with maximal contribution is picked (line 5). Then, the contribution of the remaining points in X to $S \cup \{p\}$ is updated (line 8), i.e., the portion of the contribution of each $q \in X$ that is dominated by p is removed. Finally, p is moved from X to S (lines 6 and 9). Lines 5 to 9 are repeated until S contains k points.

Note that, as with the gHSSD, ties may occur as more than one point may have the (same) highest contribution. As before, it is acceptable to choose one of them arbitrarily even though that may influence the greedy solution obtained.

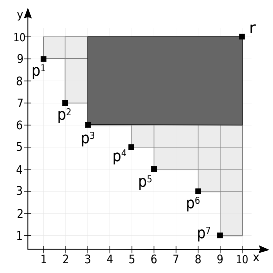
Algorithm 4.6 reduces to the consecutive computation of several incremental cases of UpdateAllContributions2 problem (lines 7 and 8). Note that although lines 2 and 3 also correspond to the AllContributions2 problem, its computation is trivial because in this case R is the empty set ($R = \{\}$). As discussed in Section 3.4.5 (see description of gHSS [D2]) neither the algorithms to compute UpdateAllContributions nor those to compute AllContributions can be applied directly to update/compute the contribution of every point $q \in X$ to S. Without specific algorithms for the AllContributions2 problem, the only viable instances of gHSS are based on computing multiple OneContribution problems, i.e., by computing, one by one, either $H(p, q, S)$ or $H(q, S \cup \{p\})$ for every $q \in X$. The current asymptotically best instances of gHSS for $d = 3$ is with UHV3D (see Table 3.5), or even with HV3D$^+$ (either -U or -R versions) proposed in Section 4.1, either way resulting in an algorithm with $O(k^2 n)$ time complexity.

4.4.2 2-Dimensional Example

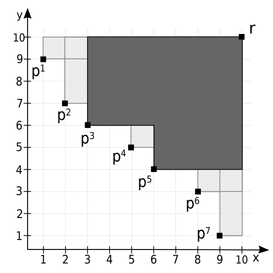
Figure 4.10 shows an example of how the greedy algorithm can be applied to a set of nondominated points $X = \{p^1, \ldots, p^7\}$ in two-dimensional space for k up to 7. Figure 4.10 illustrates which points are selected by the greedy algorithm and in which order. Figures 4.10(a)-(d) show the evolution of sets X and S in the first 4 iterations,



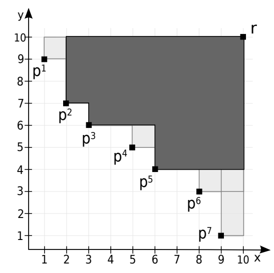
(a) $S = \{\}$
$X = \{p^1, \dots, p^7\}$



(b) $S = \{p^3\}$
$X = \{p^1, p^2, p^4, \dots, p^7\}$



(c) $S = \{p^3, p^5\}$
$X = \{p^1, p^2, p^4, p^6, p^7\}$



(d) $S = \{p^3, p^5, p^2\}$
$X = \{p^1, p^4, p^6, p^7\}$

		|S|						
	$H(p, S)$	0	1	2	3	4	5	6
p	p^1	9	2	2	1	**1**	-	-
	p^2	24	3	**3**	-	-	-	-
	p^3	**28**	-	-	-	-	-	-
	p^4	25	5	1	1	1	**1**	-
	p^5	24	**8**	-	-	-	-	-
	p^6	14	6	2	2	1	1	**1**
	p^7	9	5	3	**3**	-	-	-

(e) Contributions with respect to S

Figure 4.10: Example of the incremental greedy algorithm (gHSS2D) for the HSSP in 2 dimensions.

and depict the corresponding contributions of the points in X to S. The table in Figure 4.10(e) shows the values of these contributions for every iteration of the algorithm. Each column corresponds to an iteration of the algorithm, and is labeled with the size of the set of selected points, |S|. Each row shows the contribution of a point in X as set S grows. Consequently, column i shows the contribution to S of each point not yet selected at iteration i. The values in bold indicate which point from X is selected in each iteration. Note that column i shows both the points in the greedy solution for $k = i$ (the dashed cells) and an intermediate solution for $k > i$. Thus, any particular case where $k < n$ leads to performing just the first k steps of the example.

The first step of the algorithm (line 3 in Algorithm 4.6) is to compute the contribution of every point $p \in X$ to $S = \{\}$, which corresponds to the first column in the table. Then, in iteration 1 of the for loop in line 4, since p^3 contributes the most to $S = \{\}$, it is selected and moved from X to S (lines 6 and 9). The contributions of the points remaining in X are updated (line 8) to account for the addition of p^3 to S. For example, the contribution of p^1 is updated by subtracting the area of its contribution that becomes dominated by p^3 (the area between $(3, 9)$ and r). Formally, $H(p^1, \{p^3\}) = H(p^1, \{\}) - H(p^1, p^3, \{\}) = 9 - 7 = 2$. The same calculations are performed for $p^2, p^4, \dots, p^7$. After the contribution update step, the contributions of every point still in X to $S = \{p^3\}$ are known as shown in the second column of the table. In iteration 2, p^5 is the point that contributes the most to $S = \{p^3\}$, and so it is selected and moved from X to S. Note that, in such a case, the contributions of p^1 and p^2 remain the same with the addition of p^5 to S, and so only those of p^4, p^6 and p^7 have to be updated. The algorithm repeats the select and update steps until k points are selected. As in the example of gHSSD, the lowest value of the x-coordinate was used as a tiebreaker (for example, in the third iteration p^2 was selected instead of p^7). In Figure 4.10, the greedy solution for $k \leq n$ is formed by the first k points of the sequence: $p^3, p^5, p^2, p^7, p^1, p^4, p^6$. For a given $k < n$, the greedy solution contains the points that, in the column k, correspond to dashed cells. For example, for $k = 3$, the greedy solution can be seen in the fourth column, which corresponds to $|\mathrm{S}| = 3$, and is $\mathrm{S} = \{p^2, p^3, p^5\}$ for which $H(\mathrm{X}) = 39$. Note that, although in this example the hypervolume indicator of the final and every intermediate solution was the same for both gHSSD and gHSS, this is not expected to happen in general.

The following sections show how $H(p, q, \mathrm{S})$ can be efficiently computed in the 2- and 3-dimensional cases.

4.4.3 Algorithm for $d = 2$

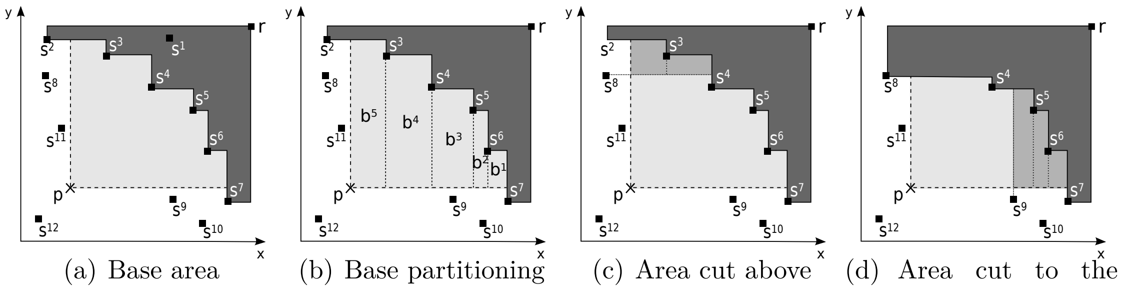
The algorithm proposed in this section (gHSS2D) deals with the particular case of gHSS in 2 dimensions. In gHSS2D, a simple procedure can be used to update the contributions of the unchosen points in line 8 of Algorithm 4.6. Assume that points in $\mathrm{X} \cup \mathrm{S}$ are kept sorted in ascending order of the y-coordinate (and in descending order of the x-coordinate). Therefore, every point $q \in \mathrm{X} \cup \mathrm{S}$ needs to keep the information of the next and of the previous point in $\mathrm{X} \cup \mathrm{S}$ which is determined once by sorting X in a pre-processing step.

Because a contribution is represented by a rectangle, every $q \in \mathrm{X}$ stores its contribution ($q.c$) and also the corresponding upper bound. All upper bounds are initially set to the reference point. Let t^1 and t^2 be the closest points in S to the left and to the right of p, respectively. Then, for each point p chosen, the points to its left are visited, in ascending order of coordinate y, until t^1 is reached. Similarly, the points to its right are visited until t^2 is reached. Every point $q \in \mathrm{X}$ between t^1 and p has the upper bound previously set to (t^2_x, t^1_y) and therefore, the quantity $H(p, q, \mathrm{S})$ is $(t^2_x - p_x) \times (t^1_y - q_y)$, and the upper bound is set to (p_x, t^1_y). Points $q \in \mathrm{X}$ between p and t^2 have $H(p, q, \mathrm{S}) = (t^2_x - q_x) \times (t^1_y - p_y)$ and their upper bound is set to (t^2_x, p_y).

The time complexity of gHSS2D is $\Theta(n(k + \log n))$ because the initial sorting costs $\Theta(n \log n)$-time and, for each point p chosen, up to n contributions have to be computed, where each contribution is computed in constant time. A worst case example is the set of n points $\{(-i, -2^{n-i+1} + 1) \mid i \in \{1, \ldots, n\}\}$ with $r = (0, 0)$. Note that gHSS2D and the exact algorithms [37, 100] have similar time complexities. However, the greedy version should be easier to implement and, because it is very simple and uses simple data structures, it is very fast in practice.

4.4.4 Computing OneContribution in $d = 3$

The efficient computation of the OneContribution problem in 3 dimensions is an important aspect of the incremental greedy algorithm for the HSSP developed in the next section. Therefore, an $O(n)$ time algorithm is explained here which can be seen as a hybrid between UHV3D (see [B5]) and the modified version of HV3D$^+$ (from Section 4.1) for the OneContribution problem briefly explained in Section 4.1.1 (see subsection on "Hypervolume Updates"). It is a simplistic version of the two,



(a) Base area (b) Base partitioning (c) Area cut above (d) Area cut to the right

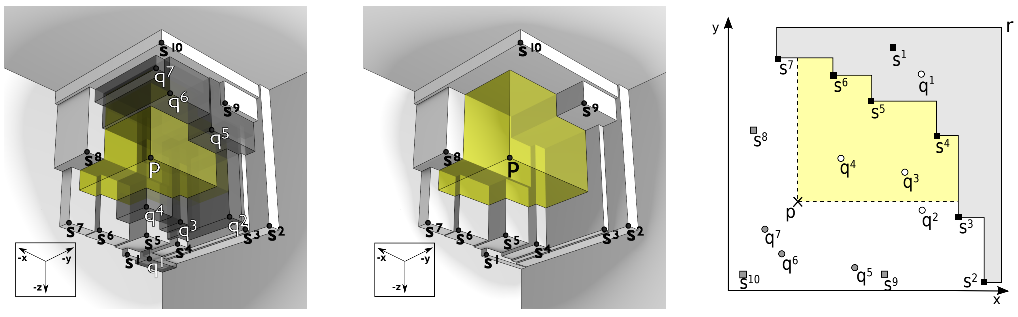
Figure 4.11: Example of the computation of the contribution $H(p, \mathrm{S})$, where $\mathrm{S} = \{s^1, \ldots, s^{12}\}$. It is assumed that $s^1_z < \cdots < s^7_z < p_z < s^8_z < \cdots < s^{12}_z$.

consisting of the sweeping of the former and the area update of the latter, leaving out the box-division of the contribution used in UHV3D and the additional information required by the data structure of HV3D$^+$ (cx and cy). The resulting algorithm for computing OneContribution in $d = 3$ is described next using Figure 4.11 for illustration. It will be referred to as IHV3D.

Given a point $p \in \mathbb{R}^3$ and a set $\mathrm{S} \subset \mathbb{R}^3$ of n points, the contribution $H(p, \mathrm{S})$ is computed in IHV3D by sweeping the points $q \in \mathrm{S}$ such that $q_z > p_z$ in ascending order of the z coordinate, and partitioning the 3-dimensional contribution in horizontal slices. The contribution of p is the sum of the volumes of all slices. The volume of a slice is the area of the base of that slice multiplied by its height. The height of a slice is the absolute difference between the two consecutive points defining that slice. The base is delimited by the projection onto the (x, y)-plane of the first point defining that slice and the points below it in z. Thus, S is split into two sets, $\mathrm{S}^1 = \{q \in \mathrm{S} \mid q_z \leq p_z\}$ and $\mathrm{S}^2 = \{q \in \mathrm{S} \mid q_z > p_z\}$. In addition, a set of points whose projections on the (x, y)-plane delimit the area exclusively dominated by p in each iteration, S', is maintained. This set of mutually nondominated points is initialized with such points in S^1 to represent the base of the first slice.

Both the splitting of S and the initialization of S' are performed in linear time. In the example of Figure 4.11(a), $\mathrm{S}^1 = \{s^1, \ldots, s^7\}$, $\mathrm{S}^2 = \{s^8, \ldots, s^{12}\}$ and $\mathrm{S}' = \{s^2, \ldots, s^7\}$. Note that at most two points in S' are not dominated by p on the (x, y)-plane, one above and to the left, and another below and to the right (s^2 and s^7 in the example).

The area of the base of the first slice is computed by adding up the areas of the non-overlapping rectangles into which the base is partitioned (see Figure 4.11(b)) as the points in S' are visited in ascending order of y. Then, the points in S^2 are visited in ascending order of z. For each new point, the volume of the current slice is computed and the area of its base is updated to obtain the base area of the next slice. In the example, the first point visited is s^8. Therefore, the area of the base of the bottom slice is multiplied by $s^8_z - p_z$. Then, the base area is updated by subtracting the area of the region that is now dominated also by s^8 (see Figure 4.11(c)). This area is computed by visiting the points in S' that are dominated by s^8 on the (x, y)-plane. In the example, these are points s^2 and s^3, which are subsequently replaced in S' by s^8. Hence, S' becomes $\mathrm{S}' = \{s^8, s^4, \ldots, s^7\}$. The procedure for s^9 is similar (see Figure 4.11(d)). Visited points that do not dominate part of the region dominated by p are skipped (e.g. s^{10}).



(a) 3D example (b) $H(\mathrm{S})$ and $H(p, \mathrm{S})$ (c) 2D projection with $H(\mathrm{S}^*)$ and $H(p^*, \mathrm{S}^*)$ at $z = p_z$

Figure 4.12: Example where $\mathrm{X} = \{q^1, \ldots, q^7\}$, $\mathrm{S} = \{s^1, \ldots, s^{10}\}$ and p is the last point removed from X and to be added to S. (a) shows X, S and p, (b) shows how much volume p will add to S (transparent region). (c) shows a cut at $z = p_z$ and the 2-dimensional projection of all points in (a). In (c), points in X are represented with circles, and points in S are represented with squares. The squares and circles painted gray have higher z coordinate than p and the remaining ones have lower z coordinate than p. As can be seen in (a), when $\mathrm{X} \cup \mathrm{S} \cup \{p\}$ is sorted in ascending order of the z coordinate, the following sequence is obtained: $q^1, s^1, \ldots, s^7, q^2, \ldots, q^4, p, s^8, q^5, q^6, s^9, q^7, s^{10}$.

The algorithm continues until a point in S^2 that dominates p on the (x, y)-plane is found, s^{12} in the example. The volume of the last slice is computed by multiplying the current base area by $(s_z^{12} - s_z^{11})$. In IHV3D, all sets are implemented as sorted lists, and sentinels are used to ensure that limiting points such as s^2, s^7 and s^{12} always exist. IHV3D has an amortized $O(n)$ time complexity because each point in S is visited once when it is added to S' and a second time when it is removed from S', and all operations on S' are performed in constant time.

4.4.5 Algorithm for $d = 3$

The algorithm that deals with the particular case of gHSS in 3 dimensions (gHSS3D) is presented in this subsection. This subsection starts by defining the data structures, some notation and procedures used by the algorithm. Then, the algorithm itself and, in particular, the update of the contribution of the points in X in line 8 of Algorithm 4.6 are detailed. This subsection finishes with a discussion of the time-complexity of gHSS3D.

The main aspect of gHSS3D is how UPDATEALLCONTRIBUTIONS2 is solved, i.e., how the contributions of points in X are updated. For an easier understanding of how those are performed in gHSS3D, the problem depicted in Figure 4.12 will be used as an example. The figure shows an intermediate iteration of Algorithm 4.6, where some points have already been moved from X to S. Note that Figure 4.12 does not intend to illustrate an actual choice of points by gHSS3D, and that the update procedure presented here is independent of the choice of points moved from X to S. The only assumptions about X and S are that they are disjoint sets and $\mathrm{X} \cup \mathrm{S}$ is a nondominated point set. In Figure 4.12(a), the contribution to S of every point in X is shown. The transparent (yellow) volume in Figure 4.12(b) shows more clearly

Algorithm 4.7 gHSS3D(X, k, r)

```
Require: $X \subset \mathbb{R}^d$, $k \in \mathbb{N}^+$, $r \in \mathbb{R}^d$
1: X is sorted in ascending order of all dimensions
2: $S \leftarrow \{(r_x, -\infty, -\infty), (-\infty, r_y, -\infty), (-\infty, -\infty, r_z)\}$
3: for all $q \in X$ do
4:   $q.c \leftarrow (r_x - p_x) \times (r_y - p_y) \times (r_z - p_z)$
5: for $i = 1..k$ do
6:   initialize $q.v$ to 0 for all $q \in X$
7:   $p \leftarrow \arg\max_{q \in X}\{q.c\}$
8:   $X \leftarrow X \backslash \{p\}$
9:   $X' \leftarrow X$
10:  for all $o \in \{(x,y,z),(z,x,y),(y,z,x)\}$ do
11:    $(x,y,z) \leftarrow o$ // Change coordinate order
12:    $X^1 \leftarrow \{q \in X' \mid p^* \leq q^*$ and $p_z \geq q_z\}$
13:    $X^2 \leftarrow \{q \in X' \mid p^* \geq q^*$ and $p_z \leq q_z\}$
14:    $S^1, S^2 \leftarrow \mathsf{split}_z(S, p_z)$
15:    jointContributions1($p, S^1, S^2, X^1, r$)
16:    jointContributions2($p, S^1, S^2, X^2, r$)
17:    $X' \leftarrow X' \backslash (X^1 \cup X^2)$
18:  for all $q \in X$ do
19:    $q.c \leftarrow q.c - q.v$
20:  $S \leftarrow S \cup \{p\}$
21: return S
```

the volume that p adds to S. Any point in X whose contribution to S lies partially in that yellow region has to have that volume removed from its contribution, as it becomes dominated. 2D projections as the one in Figure 4.12(c) will be used further in this paper.

In the context of gHSS3D, the procedure to solve the UPDATEALLCONTRIBUTIONS2 problem is explained just for the incremental scenario. However, since it consists of computing joint contributions, the decremental scenario is similar.

Data structures and procedures

In gHSS3D, doubly linked lists are used to maintain the sets of points sorted, and sentinels ensure that there is always a point in the limiting conditions. Algorithm 4.7 keeps both sets X and S sorted in ascending order of all coordinates. Each point $q \in X$ keeps some information associated to it, such as area ($q.a$), volume ($q.v$), height ($q.z$) and contribution ($q.c$). The first three values are temporary values that are used to compute the volume $H(p, q, S)$ which will be subtracted from $q.c$, the contribution of q. The value $q.z$ indicates the value of the third coordinate up to which volume $q.v$ has been updated, and $q.a$ keeps the area dominated at height $q.z$. p^* and S^* will denote the projections of p and S onto the (x, y)-plane, respectively.

Given $X, S \subset \mathbb{R}^3$ represented by sorted lists and the points $p, q \in \mathbb{R}^3$ and $h \in \mathbb{R}$, the following procedures are available.

next$_y(p, S)$ The point following p in S with respect to coordinate y, for $p \in S$.

head$_y(S)$ The point $q \in S$ with the least q_y.

min$_y(p, \mathrm{S})$ The point $q \in \mathrm{S}$ with the least $q_y > p_y$ such that $q_x \leq p_x$.

addFirst$_y(p, \mathrm{S})$ Add point p to S where p becomes head$_y(\mathrm{S})$.

minima(S) Return the points that are not dominated on the (x, y)-plane , i.e., $\{q \in \mathrm{S} \mid \nexists t \in \mathrm{S} : \ t^* \leq q^*, t \neq q\}$.

split$_y(\mathrm{S}, h)$ Given $h \in \mathbb{R}$, split S in two sets, S^1 and S^2 such that $\mathrm{S}^1 = \{q \in \mathrm{S} \mid q_y \leq h\}$ and $\mathrm{S}^2 = \{q \in \mathrm{S} \mid q_y > h\}$ and return S^1 and S^2.

area(p, S) The quantity $H(p^*, \mathrm{S}^*)$.

updateVolume(X, h) Given $h \in \mathbb{R}$, for each point $q \in \mathrm{X}$, update its volume, save it in $q.v$ and update $q.z$, i.e., compute $q.v \leftarrow q.v + q.a \times (h - q_z)$ and set $q.z \leftarrow h$.

initializeBases$(\mathrm{X}, p, \mathrm{S})$ For each point $q \in \mathrm{X}$, compute $H(p^*, q^*, \mathrm{S}^*)$ and save it in $q.a$.

updateAreas$(p, \mathrm{X}, \mathrm{S})$ For each $q \in \mathrm{X}$ compute $q.a \leftarrow q.a - H(p^*, q^*, \mathrm{S}^*)$.

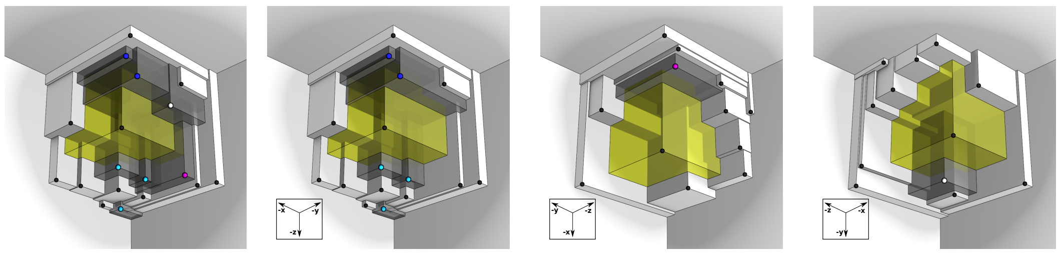
Procedures next, head, min, addFirst and split are also available for coordinates x and z and, apart from split and min, they all run in constant time. split$_y$ has a cost of $O(|\mathrm{S}^2|)$ because it is the cost of finding the break point by sweeping points in descending order of coordinate y. The remaining procedures have linear cost w.r.t. the total size of the input sets, i.e., either $O(|\mathrm{X}|)$, $O(|\mathrm{S}|)$ or $O(|\mathrm{S}| + |\mathrm{X}|)$. Both initializeBases and updateAreas will be explained in more detail in Subsection 4.4.3. All procedures that modify or return a subset of a given sorted set guarantee that the returned sets are also sorted according to the coordinate used for sweeping the points. These procedures may also guarantee that those points are sorted according to other coordinates, if needed. More information about those procedures will be given in the next subsections. Note that, if a set of nondominated points on the (x, y)-plane is sorted in ascending order of one coordinate, then it is also sorted according to the other coordinate, but in descending order.

Main loop of gHSS3D

gHSS3D follows the same working principle as gHSS, but, instead of updating the contributions of points in X one by one (lines 4 and 8 in Algorithm 4.6), the (x, y, z)-space is divided into 8 octants with a common vertex at p, and the contributions of the points in each pair of opposite octants are updated at the same time (lines 9 to 17 in Algorithm 4.7).

The two octants corresponding to the region that dominates p and the region dominated by p are ignored because they do not contain any points. The remaining three pairs of octants are all updated in the same way, except that a different coordinate order is considered for each pair. The order is set in such a way that a different dimension is used as the z-coordinate in each case (lines 10-17 of Algorithm 4.7).

Given a coordinate order $o \in \{(x, y, z), (z, x, y), (y, z, x)\}$, the two octants considered when order o is selected are those that contain sets $\mathrm{X}^1 \subseteq \mathrm{X}$ and $\mathrm{X}^2 \subseteq \mathrm{X}$, defined as follows. X^1 contains the points in X that are dominated by p in the first two dimensions of o, but are equal to or better (i.e., lower) than p in the third dimension. X^2 contains the points in X that dominate p in the first two dimensions



(a) Base example (b) $o = (x, y, z)$, $X^1 = \{q^1, q^3, q^4\}$, $X^2 = \{q^6, q^7\}$ (c) $o = (y, z, x)$, $X^1 = \{\}$, $X^2 = \{q^2\}$ (d) $o = (z, x, y)$, $X^1 = \{q^5\}$, $X^2 = \{\}$

Figure 4.13: The subproblems X^1 and X^2 for the three objective orders considered, $o = \{(x, y, z), (y, z, x), (z, x, y)\}$.

of o but are equal to or worse (i.e., higher) than p in the third dimension. Figure 4.13 shows how the set X from the example in Figure 4.12 is split into octants, and shows X^1 and X^2 according to the objective order considered. In Figure 4.13(a), points are assigned colors, each associated with a single octant. Figures 4.13(b), 4.13(c) and 4.13(d) show the corresponding sets X^1 and X^2 according to objective order (x, y, z), (y, z, x) and (z, x, y), respectively. Note that considering a different objective order is equivalent to rotating the space.

Lines 9 and 17 of Algorithm 4.7 guarantee that no point in X is updated more than once in case there are points with repeated coordinates, i.e., points on the boundary between two octants. The computation of $q.v = H(p, q, S)$ for $q \in X^1$ and $q \in X^2$ is detailed in Algorithms 4.9 and 4.8, respectively.

Updating contributions of points in X

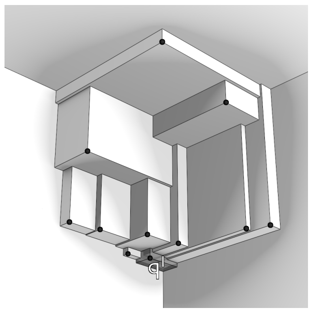
Consider the case where the order considered is $o = (x, y, z)$. In the example given, $X^1 = \{q^1, q^3, q^4\}$ and $X^2 = \{q^6, q^7\}$, as depicted in Figure 4.13(b). The update procedure for the remaining coordinate orders is similar (Figures 4.13(c) and 4.13(d)). Figure 4.14 shows (in red) the joint contributions of p with each point in X^1 (Figures 4.14(a) to 4.14(c)) and with each point in X^2 (Figures 4.14(d) and 4.14(e)), which have to be computed and removed. Note that, in the case of p^1, there is no joint contribution with p. Furthermore, note that some of the joint contributions (partially) overlap, for example, those of p with q^3 and of p with q^4.

The update procedure will be explained first for X^2 and then for X^1. In both cases, the algorithms are built upon IHV3D (explained in Section 4.4.4). Therefore, Algorithm 4.7 splits S into S^1 and S^2. In this case only, function split takes $O(n)$-time as it has to guarantee that S^1 and S^2 are sorted according to all dimensions. Set S' is also maintained along the execution of both Algorithms 4.9 and 4.8, and is initialized as the subset of S^1 that delimits the area dominated by p. In the example, we have that $S^1 = \{s^1, \ldots, s^7\}$, $S^2 = \{s^8, \ldots, s^{10}\}$ and $S' = \{s^3, \ldots, s^7\}$ (initial set).

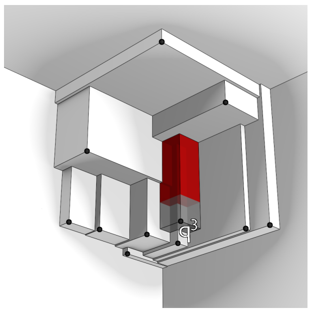
In the case of X^2 ($X^2 = \{q^6, q^7\}$), all points dominate p on the (x, y)-plane, and have higher z-coordinate. Therefore, if X^2 is sorted in ascending order of z, then, given any $q \in X^2$ and its previous point u in X^2, the joint contribution of q with p is equal to the joint contribution of u with p above the value q_z of coordinate z. Moreover, the contribution of q when it is $\mathsf{head}_z(X^2)$ is equal to the contribution of p above the value q_z of coordinate z. Figure 4.15(a) shows the joint contributions

Algorithm 4.8 jointContributions2($p, \mathrm{S}^1, \mathrm{S}^2, \mathrm{X}^2, r$)

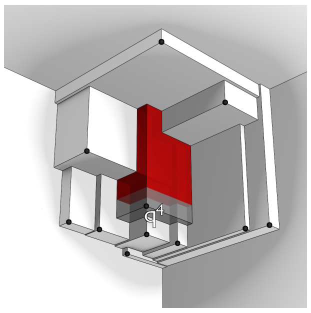
```
1: $\mathrm{S}' \leftarrow \{q \in \mathsf{minima}(\mathrm{S}^1) \mid p^* \leq q^*\} \cup \{\min_x(\mathrm{S}^1), \min_y(\mathrm{S}^1)\}$
2: $p.a \leftarrow \mathsf{area}(p, \mathrm{S}')$
3: $u \leftarrow p$
4: $\mathrm{L} \leftarrow \mathrm{X}^2 \cup \mathrm{S}^2$
5: $q \leftarrow \mathsf{head}_z(\mathrm{L})$
6: while $q \in \mathrm{X}^2$ or $q^* \nleq p^*$ do
7:    if $q \in \mathrm{S}^2$ then
8:       if $q_x \leq p_x$ then
9:          $\mathrm{S}', \mathrm{T} \leftarrow \mathsf{split}_y(\mathrm{S}', q_y)$
10:         $p.a \leftarrow p.a - \mathsf{area}(p \vee q, \mathrm{T} \cup \{\mathsf{head}_x(\mathrm{S}')\})$
11:         $\mathsf{addFirst}_x(q, \mathrm{S}')$
12:      else
13:         $\mathrm{S}', \mathrm{T} \leftarrow \mathsf{split}_x(\mathrm{S}', q_x)$
14:         $p.a \leftarrow p.a - \mathsf{area}(p \vee q, \mathrm{T} \cup \{\mathsf{head}_y(\mathrm{S}')\})$
15:         $\mathsf{addFirst}_y(q, \mathrm{S}')$
16:      $\mathsf{updateVolume}(\{u\}, q_z)$
17:      $u.a \leftarrow p.a$
18:   else
19:      $\mathsf{updateVolume}(\{u\}, q_z)$
20:      $q.a \leftarrow p.a$
21:      $q.z \leftarrow q_z$
22:      $q.prev \leftarrow u$
23:      $u \leftarrow q$
24:   $q \leftarrow \mathsf{next}_z(q, \mathrm{L})$
25: $\mathsf{updateVolume}(\{u\}, q_z)$
26: $vol \leftarrow 0$
27: while $u \neq p$ do
28:   $vol \leftarrow vol + u.v$
29:   $u.v \leftarrow vol$
30:   $u \leftarrow u.prev$
```



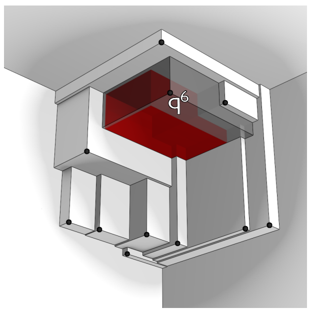
(a) $H(q^1, \mathrm{S})$



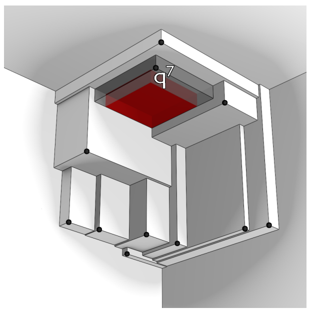
(b) $H(q^3, \mathrm{S})$



(c) $H(q^4, \mathrm{S})$



(d) $H(q^6, \mathrm{S})$



(e) $H(q^7, \mathrm{S})$

Figure 4.14: Individual contributions to S (dark gray plus red region) of the points in $\mathrm{X}^1 = \{q^1, q^3, q^4\}$ and in $\mathrm{X}^2 = \{q^6, q^7\}$. The region of each contribution that is dominated by p is in red.

of p with each point in $\mathrm{X}^2 = \{q^6, q^7\}$. Note that the joint contribution of q^7 with p is equal to the joint contribution of q^6 with p for $z \geq q^7_z$. The joint contribution of q^6 with p is equal to the contribution of p for $z \geq q^6_z$. Thus, if the volume between $z = u_z$ and $z = q_z$ is associated only to u, then the joint contribution of every $q \in \mathrm{X}^2$ can be calculated by accumulating the volume associated to each point while visiting X^2 in descending order of z. Hence, in Algorithm 4.8, the previous point $u \in \mathrm{X}^2$ of each $q \in \mathrm{X}^2$ is stored in $q.prev$.

Algorithm 4.8 behaves just like IHV3D, i.e., in the algorithm, S' is used to maintain the points delimiting the area of p at height $z = q_z$, and points in S^2 are visited in ascending order of z in order to update the area of p and to compute the slice volume (lines 8 to 15). Additionally, X^2 is swept along with S^2 in ascending order of the z-coordinate by merging the two lists (line 4). The last point visited from X^2 is recorded as u (except for first initialization of u, which is a sentinel), and its volume (line 16) and area (line 17) are updated when the area of p ($p.a$) is updated. Whenever the current point $q \in \mathrm{X}^2 \cup \mathrm{S}^2$ belongs to X^2, the volume simultaneously dominated by u and p bounded below by u_z becomes bounded above by q_z (line 19) and stays associated to (and only to) u. The contribution of p above q_z becomes associated to q (lines 20 and 21) and q is set as the new u (line 23). In that way, each part of the contribution p (a consecutive set of slices) is associated to a single point of X^2. When the first point from S^2 that dominates p on the (x, y)-plane is found (s^{10}), the volume of the last point from X^2 is updated (line 25). Figures 4.15(b) and 4.15(c) show the two slices (whose volume is) associated to q^6, and Figure 4.15(d) shows the only slice associated to q^7. Lastly, $q \in \mathrm{X}^2$ are visited again but in descending order of the z-coordinate, and their volumes are

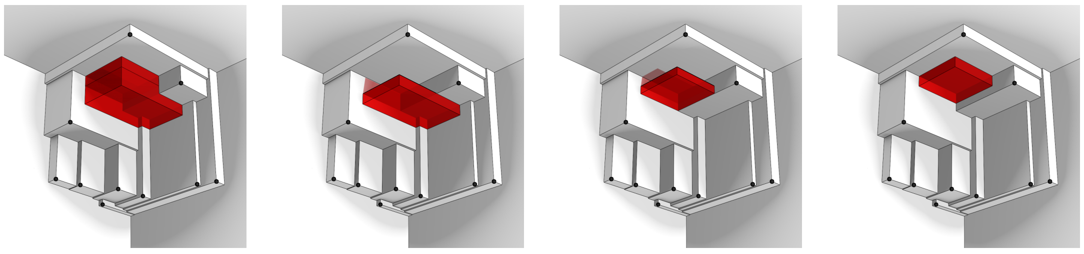

(a) $H(p, q^6, \mathrm{S})$ and $H(p, q^7, \mathrm{S})$ (b) Slice 1 (c) Slice 2 (d) Slice 3

Figure 4.15: The (partially overlapping) joint contributions of every point in $\mathrm{X}^2 = \{p^6, p^7\}$ with p and their division in slices.

accumulated and added to the next visited points (lines 27 to 30). In the example, the volume associated to q^6 is stored in *vol*, and corresponds to its joint contribution with p. Then the volume associated to q^7 is added to *vol*, corresponding now to the joint contribution of q^7 and p.

In the case of points $q \in \mathrm{X}^1$, not every point has its contribution reduced with the addition of p to S, as it is the case of q^1 in Figure 4.14(a). In Algorithm 4.9, a set $\mathrm{X}' \subseteq \mathrm{X}^1$ is used to keep the points of X^1 whose joint contributions with p are still being computed. X' is initialized in linear time with the points in X^1 that need to be updated, i.e., those points such that $q^* \in \mathrm{X}^{1*}$ is not dominated by S'^*. Figure 4.16(a) shows the joint contributions of p with every point in the initial set $\mathrm{X}' = \{q^3, q^4\}$, which partially overlap. Figures 4.16(b) to 4.16(d) show how these joint contributions are split into slices by the algorithm. The computation of $H(p, q, \mathrm{S})$ for each $q \in \mathrm{X}'$ is performed in two main steps, the computation of the base area of the volume $H(q \vee p, \mathrm{S}')$ (line 3) and the area updates (lines 13 and 21). Figures 4.17 and 4.18 will be used as examples of these steps, respectively. Note that the base areas of the volumes $H(q \vee p, \mathrm{S}')$ correspond to the base areas in the red volume in Figure 4.16(b), while the base areas of the red volumes of Figures 4.16(c) and 4.16(b) correspond to the subsequently updated areas.

Figure 4.17(a) corresponds to Figure 4.16(b) in the cut plane at $z = p_z$ and shows, also in red, the base areas of q^3 and q^4 to be computed, which overlap partially. The base areas are initialized in linear time by computing $H(p^*, \mathrm{S}'^*) - H(p^*, \mathrm{S}'^* \cup \{q^*\})$, for each $q \in \mathrm{X}'$. The areas associated to points in X' are initialized with $H(p^*, \mathrm{S}'^*)$, which was previously computed in linear time in line 2 of Algorithm 4.8. The value $H(p^*, \mathrm{S}'^* \cup \{q^*\})$ is computed in two sweeps. Points $q \in \mathrm{S}' \cup \mathrm{X}'$ are first visited in ascending order of x and the area dominated by p between p_x and each q_x is computed in a cumulative way. When a point from X' is visited, the area accumulated so far is subtracted from the area associated with that point. In Figures 4.17(b) and 4.17(c), the striped area is the accumulated area that is subtracted from $q^4.a$ and $q^3.a$, respectively. The remaining area to be subtracted, i.e., the area between p_y and q_y to the right of p_x, is computed in an analogous way by sweeping points in $\mathrm{S}' \cup \mathrm{X}'$ in ascending order of the y-coordinate. Note that, in the second sweep, it is also necessary to add to $q.a$ the area bounded below by p and above by q, for each $q \in \mathrm{X}'$, as it was subtracted twice.

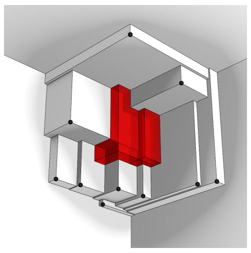
Figure 4.18 shows the update of the areas of points in X' (lines 13 and 21). The volumes are updated in lines 12 and 20. This corresponds to the cases where a point

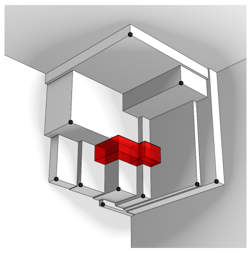
Algorithm 4.9 jointContributions1$(p, \mathrm{S}^1, \mathrm{S}^2, \mathrm{X}^1, r)$

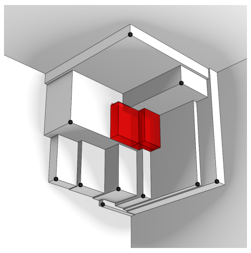
```
1: $\mathrm{S}' \leftarrow \{q \in \mathsf{minima}(\mathrm{S}^1) \mid p^* \leq q^*\} \cup \{\min_x(\mathrm{S}^1), \min_y(\mathrm{S}^1)\}$
2: $\mathrm{X}' \leftarrow \{q \in \mathrm{X}^1 \mid p^* \leq q^*$ **and** $\nexists t \in \mathrm{S}' : t^* \leq q^*\}$
3: initializeBases$(\mathrm{X}', p, \mathrm{S}')$
4: $\mathrm{L} \leftarrow \mathrm{S}^2$
5: $q \leftarrow \mathsf{head}_z(\mathrm{L})$
6: **while** $q^* \nleq p^*$ **do**
7:     **if** $q_x \leq p_x$ **then**
8:         $\mathrm{S}', \mathrm{T} \leftarrow \mathsf{split}_y(\mathrm{S}', q_y)$
9:         $t \leftarrow \mathsf{head}_x(\mathrm{S}')$
10:        $\mathrm{X}', \mathrm{A} \leftarrow \mathsf{split}_y(\mathrm{X}', q_y)$
11:        $\mathrm{B} \leftarrow \{u \in \mathrm{X}' \mid u_x < t_x\}$
12:        updateVolume$(\mathrm{A} \cup \mathrm{B}, q_z)$
13:        updateAreas$_y(p \vee q, \mathrm{B}, \mathrm{T} \cup \{\mathsf{head}_x(\mathrm{S}')\})$
14:        addFirst$_x(q, \mathrm{S}')$
15:    **else**
16:        $\mathrm{S}', \mathrm{T} \leftarrow \mathsf{split}_x(\mathrm{S}', q_x)$
17:        $t \leftarrow \mathsf{head}_y(\mathrm{S}')$
18:        $\mathrm{X}', \mathrm{A} \leftarrow \mathsf{split}_x(\mathrm{X}', q_x)$
19:        $\mathrm{B} \leftarrow \{u \in \mathrm{X}' \mid u_y < t_y\}$
20:        updateVolume$(\mathrm{A} \cup \mathrm{B}, q_z)$
21:        updateAreas$_y(p \vee q, \mathrm{B}, \mathrm{T} \cup \{\mathsf{head}_y(\mathrm{S}')\})$
22:        addFirst$_y(q, \mathrm{S}')$
23:    $q \leftarrow \mathsf{next}_z(q, \mathrm{L})$
```

in S′ cuts above the area dominated by p (lines 7 to 14 and s^8 in Figures 4.18(a) and 4.18(b)) or to the right (line 16 to 22 and s^9 in Figure 4.18(c) and 4.18(d)). Points $q \in \mathrm{S}^2$ are visited in ascending order of z-coordinate, as in IHV3D. Looking at the case where $q = s^8$, let T be the set of points in S′ with higher y-coordinate than s^8_y ($\mathrm{T} = \{s^5, \ldots, s^7\}$). In Algorithm 4.9, set T is first removed from S′ ($\mathrm{S}' = \{s^3, \ldots, s^7\}$), in $O(|\mathrm{T}|)$-time, through function split. Thus, S′ becomes $\mathrm{S}' = \{s^3, s^4\}$). T is kept sorted according to the y-coordinate. Similarly, set A is removed from X′ ($\mathrm{X}' = \{q^3, q^4\}$) in $O(|\mathrm{A}|)$-time, also through split (line 10). Set A contains the points that are dominated by q^* on the (x, y)-plane, i.e., those that have no more contribution above q_z ($\mathrm{A} = \{\}$). Finally, the areas associated with the points in X′ to the left of the point that delimits q at right ($t = s^4$) have to be updated. Therefore, those points are stored in B (line 11, $\mathrm{B} = \{q^3, q^4\}$). The volumes of points in $\mathrm{B} \cup \mathrm{A}$ are updated in $O(|\mathrm{A}| + |\mathrm{B}|)$-time (line 12). Then, in procedure updateAreas (line 13), points $u \in \mathrm{B} \cup \mathrm{T}$ are visited in descending order of coordinate x. For each u visited, the area dominated by $p \vee q$ to the right of u is accumulated. Whenever the visited point u belongs to B, the area computed so far is subtracted from $u.a$. Therefore, the update of the areas of points in B costs $O(|\mathrm{B}| + |\mathrm{T}|)$. At last, q is added to S′ ($\mathrm{S}' = \{s^8, s^4, s^3\}$). The updated areas (and the updated X′ and S′) are depicted in Figure 4.17(b), which corresponds to Figure 4.16(c) at the cut plane $z = s^8_z$.

When $q_y \leq p_y$, the procedure is analogous. Note that, in the example of Figure 4.18(c), $q = s^9$ and $\mathrm{A} = \{q^3\}$. Therefore, q^3 is removed from X′ and its volume is updated. X′ becomes $\{q^4\}$, then $t = s^8$, $\mathrm{B} = \{q^4\}$, and, at the end of that iteration,


(a) $H(p, q^3, \mathrm{S})$ and $H(p, q^4, \mathrm{S})$


(b) Slice 1


(c) Slice 2

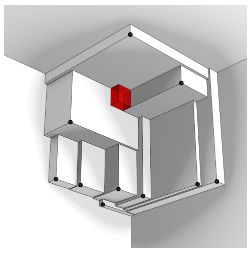
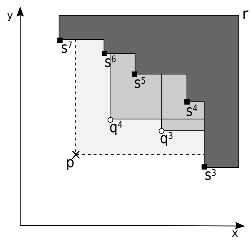
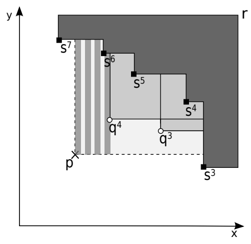

(d) Slice 3

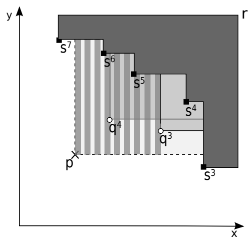
Figure 4.16: The (partially overlapping) joint contributions of every point in $\{p^3, p^4\} \subset \mathrm{X}^1$ with p and their division in slices.



(a) Areas of q^3 and q^4



(b) Update $q^4.a$



(c) Update $q^3.a$

Figure 4.17: Example of how the base area of each points in $\mathrm{X}' = \{p^3, p^4\}$ is computed in $\mathsf{initializeBases}(\mathrm{X}', p, \mathrm{S}')$.

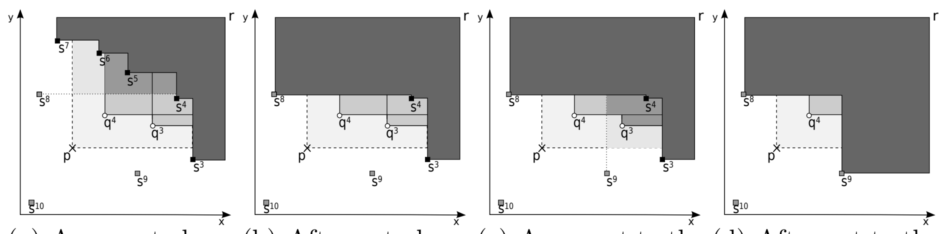
$\mathrm{S}' = \{s^8, s^9\}$ and $\mathrm{X}' = \{q^4\}$. The updated areas (and the updated X' and S') are depicted in Figure 4.18(d), which corresponds to Figure 4.16(d) at the cut plane $z = s^9_z$. Algorithm 4.9 terminates when the first point q that dominates p on the (x, y)-plane is found (s^{10}).

Time-complexity

In Algorithm 4.7 (gHSS3D), points are sorted in $O(n \log n)$-time and their contributions are initialized in linear time. Then, the contributions of points in X are updated k times by sweeping those points considering three different coordinate orders. For each coordinate order, the points of two disjoint sets are updated, X^1 and X^2 in Algorithms 4.9 and 4.8, respectively. Lines 11 to 14 and 17 have linear cost, while Algorithm 4.8 (line 16) has amortized linear time. Algorithm 4.9 (line 15) has amortized $O(nk)$-time, but the amortization is along the entire execution of Algorithm 4.7.

Algorithm 4.8 inherits the $O(n)$ time complexity of IHV3D. As in IHV3D, the base area associated with p is computed in linear time (line 2). Afterwards, points in S are only visited once when they are added to S' (line 1, 11 or 15), and once again if they are removed from S' (lines 9 and 13) and are used to update the area of p (lines 10 and 14). Points in X^2 are used in constant time operations in the first while loop (lines 16 to 25), and are all visited once again, to update their volumes (lines 27 to 30).

The time complexity of Algorithm 4.9 is analyzed by considering all calls (a



(a) Areas cut above (b) After cut above (c) Areas cut to the right (d) After cut to the right

Figure 4.18: Example of how points from S^2 are used in Algorithm 4.9 to update the areas of points in X^1.

maximum of $k \leq n$) from Algorithm 4.7. The analysis shows that some steps amortize to $O(n)$-time for each call of Algorithm 4.9, while others are worse than linear in a single call, but result in $O(nk)$-time across the k possible calls. In the worst scenario of the latter case (a single call), each point in the final set S (the greedy solution) may be compared against all other points a constant number of times.

In Algorithm 4.9, it is guaranteed that each point is visited only if its area or volume has to be updated or if it delimits the contribution of the area being computed. The operations performed to maintain S' (lines 8, 16, 14 and 22) are the same as in Algorithm 4.8, so these operations amortize to linear time. As with S', points from X^1 are added once to X' (line 2) and are removed once from X' (line 10 or 18). Therefore, maintaining X' also amortizes to linear time. It remains to explain how many times points in $\mathrm{B} \subset \mathrm{X}'$ are visited. Constructing B has a cost $O(|\mathrm{B}|)$, and procedure updateAreas has a cost of $O(|\mathrm{B}| + |\mathrm{T}|)$. Since the points in T were previously removed from S', they are used by updateAreas only once. However, in the worst case, B is equal to X' in every iteration of Algorithm 4.9, and points are never removed from X'. This means that procedure updateAreas has a cost of $O(n)$, leading to $O(kn)$-time complexity over $|\mathrm{S}^2| \leq k$ iterations.

Basically, the role of updateAreas is to compare each point in X with points in S. However, in this procedure, each point in X is compared against each point in S once throughout the execution of gHSS3D. As $|\mathrm{S}| \leq k$ always holds, all executions of updateAreas, and of updateVolume, amortize to $O(nk)$. This can be checked by noting that a point $u \in \mathrm{X}^1 \subseteq \mathrm{X}$ is only compared to a point $s \in \mathrm{S}^2 \subset \mathrm{S}$ if s delimits the contribution of u. These points are compared only because the region where s delimits the contribution of u is dominated by p. Therefore, after p is added to S, the point s will not delimit the contribution of u anymore. This is because $p \leq (u \vee s)$ and $p_z < s_z$, which means that u has no contribution to $\mathrm{S} \cup \{p\}$ above the value p_z of coordinate z ($H(u, s, (\mathrm{S} \cup \{p\}) \backslash \{s\}) = 0$). Moreover, note that $(p \vee u) < (s \vee u)$, and, thus, according to Definition 3.5 i.e., the definition of delimiter, s is not a delimiter of the contribution of u to $\mathrm{S} \cup \{p\}$. Consequently, u will not be compared again with s. In the example of Figure 4.18, neither q^3 nor q^4 will be compared against s^8 or s^9 in any future execution of procedure updateAreas. In the examples of Figures 4.14(b) and 4.14(c) it can be seen that, after removing the part of the point contribution in red, the remaining volume, in dark gray, is delimited by neither s^8 nor s^9. Therefore, Algorithm 4.7 has amortized $O(nk)$-time complexity.

Although gHSS3D has $O(n(k + \log n))$-time complexity, if it is used repeatedly for a fixed k and a growing set of n solutions by iteratively adding new small sets of $m \ll n$ solutions, then these sets may be previously sorted in $O(m \log m)$, and inserted in the data structures (linked lists) in $O(n + m)$ time. Therefore, the set of k solutions selected can be updated in $O((n + m)k + m \log m)$ time in such a scenario.

Finally, at the core of gHSS3D (and of gHSS2D) is a procedure, relevant by itself, for the UpdateAllContributions2 problem. Such an update procedure is well suited for both incremental and decremental scenarios of such a problem. A single call has $O(|\mathrm{S}| \cdot |\mathrm{R}|)$ time complexity (where S is the set of points whose contributions to R have to be updated). When used sequentially for the incremental scenario (as in gHSS) the number of operations amortize to linear time per point added to R.

4.4.6 No Approximation Ratio for the HSSPComplement

The incremental greedy approach provides an approximation bound for the HSSP. However, theoretical results on the approximation to the HSSPComplement are currently unknown. In fact, similarly to the decremental greedy approach, there is no approximation bound to the HSSPComplement using an incremental greedy approach, i.e., the ratio of the contribution of the points left out to the set of chosen points between the greedy and the optimal solution, can be arbitrarily large:

Theorem 4.1. *For all $d \geq 2$, and for all $\varepsilon \geq 1$, there is a point set $\mathrm{X} \subset \mathbb{R}^d$ where $|\mathrm{X}| = n = 3$, for which the incremental greedy and optimal subsets of* HSSP *given size $k = 2$, S^g and S^{opt}, respectively, are such that*

$$\frac{H(\mathrm{X}) - H(\mathrm{S}^g)}{H(\mathrm{X}) - H(\mathrm{S}^{opt})} = \varepsilon \tag{4.2}$$

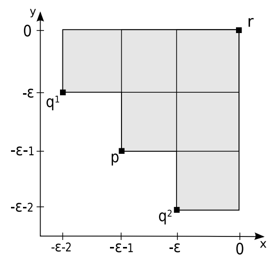


Figure 4.19: Example of a bad case for gHSS.

Proof. The theorem is proved with an example for $d = 2$, and then it is extended for the general case of $d \geq 2$. Let $r = (0, 0)$ be the reference point, and let $\mathrm{X} = \{p, q^1, q^2\}$ be such that:

$$\begin{aligned} p &= (-\varepsilon - 1, -\varepsilon - 1) \\ q^1 &= (-\varepsilon - 2, -\varepsilon) \\ q^2 &= (-\varepsilon, -\varepsilon - 2) \end{aligned}$$

Set X is depicted in Figure 4.19. For $\varepsilon \geq 1$, the following holds: $H(\{q^1, p\}) = H(\{q^2, p\}) \leq H(\{q^1, q^2\})$. Thus, $\mathrm{S}^{\mathrm{opt}} = \{q^1, q^2\}$ is an optimal subset for $k = 2$. In gHSS, p is selected first as it contributes the most to the empty set and, as q^1 and q^2 contribute equally to $\{p\}$, let us assume that q^1 is chosen next and so $\mathrm{S}^{\mathrm{g}} = \{p, q^1\}$. Therefore:

$$H(\mathrm{S}^{\mathrm{opt}}) = \varepsilon^2 + 4\varepsilon \tag{4.3}$$

$$H(\mathrm{S}^{\mathrm{g}}) = \varepsilon^2 + 3\varepsilon + 1 \tag{4.4}$$

$$H(\mathrm{X}) = \varepsilon^2 + 4\varepsilon + 1 \tag{4.5}$$

Then, the result in Theorem is deduced for $n = 3$ and $d = 2$:

$$\frac{H(\mathrm{X}) - H(\mathrm{S}^{\mathrm{g}})}{H(\mathrm{X}) - H(\mathrm{S}^{\mathrm{opt}})} = \frac{\varepsilon^2 + 4\varepsilon + 1 - \varepsilon^2 - 3\varepsilon - 1}{\varepsilon^2 + 4\varepsilon + 1 - \varepsilon^2 - 4\varepsilon} = \frac{\varepsilon}{1} \tag{4.6}$$

For the general case of $d \geq 2$, it is enough to set $r = (0, \ldots, 0)$ and set the remaining coordinates of points in X to -1 as it does not influence the calculations. □

4.5 Comparison of Greedy Algorithms for the HSSP

Different aspects of the incremental and decremental greedy approaches are analyzed in this section. The aim is to show that both approaches are able to find subsets with good approximation ratio, far from the theoretical bound, that are visually well-distributed and similar to the optimal subset. The aim is also to show that the greedy solutions are computed in a short amount of time, and to determine in which settings of n and k each of the greedy algorithms proposed is more advantageous. Therefore, the analysis is divided in three parts. The first is an analysis of the approximation results (see Figures 4.20 to 4.23 and Tables 4.1 and 4.2), the second is an analysis of the best subsets found with each approach (see Figures 4.24 to 4.27) and the last one is an analysis of the algorithms' runtime (see Figures 4.29 and 4.30).

The proposed algorithms, gHSS2D, gHSS3D[6] and gHSSD3D[7], were implemented in C and compiled with gcc version 4.7.2. All tests were run on an Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz with 6 MB of cache and 8 GB of RAM. To obtain the exact solutions for the HSSP, a Java implementation of HypSSP (see HypSSP [C3] in Section 3.4.4) was used for the 2-dimensional case. Regarding the 3-dimensional case, optimal results were obtained with the HSSP Integer Linear Programming (ILP) formulation (see ILP [C4] in Section 3.4.4), and solved with the GNU Linear Programming Toolkit.

The algorithms were tested with the following data sets (see Section 3.5): the linear front, the (spherical) convex front and the (spherical) concave front both for the 2 and 3-dimensional cases. Additionally, wave fronts with $1, \ldots, 5$ and 10 convex regions were used in the 2-dimensional case and the degenerate front was used in the 3-dimensional case. The reference point was set to $(1, 1)$ and to $(1, 1, 1)$ in the two- and the three-dimensional cases, respectively. Plots will be used to show some example data sets and results on all runs are summarized in tables.

[6]The source code of gHSS2D and gHSS3D algorithms is available at `https://github.com/apguerreiro/gHSS` (version 1.1 [79] was the one used).

[7]The source code is available at `https://github.com/apguerreiro/HVC`.

4.5.1 Approximation Quality

The evaluation of the approximation quality of the greedy solutions will be based on two ratios, the *hypervolume ratio* and the *uncovered hypervolume ratio*. The former is concerned with the approximation to the HSSP and the latter to the HSSPComplement. Given a point set $X \subset \mathbb{R}^d$ such that $|X| = n$, let $S^g \subseteq X$ and $S^{opt} \subseteq X$ be the greedy and the optimal k-sized subsets for the HSSP, respectively. Given a reference point, the hypervolume ratio is given by:

$$\frac{H(S^g)}{H(S^{opt})} \tag{4.7}$$

which is at most 1. The unconvered hypervolume ratio is given by:

$$\frac{H(X) - H(S^g)}{H(X) - H(S^{opt})} \tag{4.8}$$

which is at least 1 and where the value $H(X) - H(S^g)$ is the uncovered hypervolume of S^g, i.e., the contribution of the set of discarded points, $X \backslash S^g$, to S^g. The hypervolume ratio is used to confirm the theoretical bounds, and to show that gHSS and gHSSD performed better than their respective approximation bounds to the HSSP in all tests. In fact, both algorithms performed better than each other's bound, as well. The uncovered hypervolume ratio is used for additional insight regarding the greedy approximations as the hypervolume ratio tends to be less informative as k and n grow. Furthermore, the hypervolume ratio is more sensitive to the reference point, the farther away is the reference point, the higher is the ratio and thus, less informative. This ratio is also used to confirm that neither gHSS nor gHSSD provide an approximation bound to the HSSPComplement.

Figures 4.20 and 4.21 show the results on the 2-dimensional and the 3-dimensional data sets, respectively, regarding the hypervolume indicator and the uncovered hypervolume of the approximate and exact solutions, for growing values of k. Additionally, Tables 4.1 and 4.2 summarize the results of all tests performed related to the two ratios for the 2- and 3-dimensional case, respectively. For the 2-dimensional case, the data sets tested had sizes $n = 10, 20, \ldots, 100, 200, \ldots, 1000$ and for the 3-dimensional case, had sizes $n = 10, \ldots, 50$, except for the linear front where $n = 10, 20$, and the convex front where $n = 10, \ldots, 40$, due to the excessive amount of time required by the ILP solver.

Let us look at the first two rows in both figures and at the left-hand side of both tables. In those, the hypervolume indicator values of the selected k-sized subsets and the hypervolume ratio show that the greedy algorithms are able to find subsets whose hypervolume is very close or equal to the optimal subsets. The hypervolume ratio between the incremental greedy and the optimal solution was always found to be greater than 0.8929, which is much better than the theoretical bound of $(1 - 1/e) \simeq 0.6321$. On the other hand, although the theoretical bound for the decremental greedy proved in Section 4.3.4 is quite pessimistic, particularly for small k, gHSSD was also always found to achieve a high hypervolume ratio which was always greater than 0.8259. Moreover, the plots show that for both greedy approaches the approximation ratio was generally lower for smaller values of k, with $k = 2$ being the subset size that most often led to the lowest approximation ratio. For $k \geq n/2$, the ratio was found to be either 1 (an optimum was found) or very

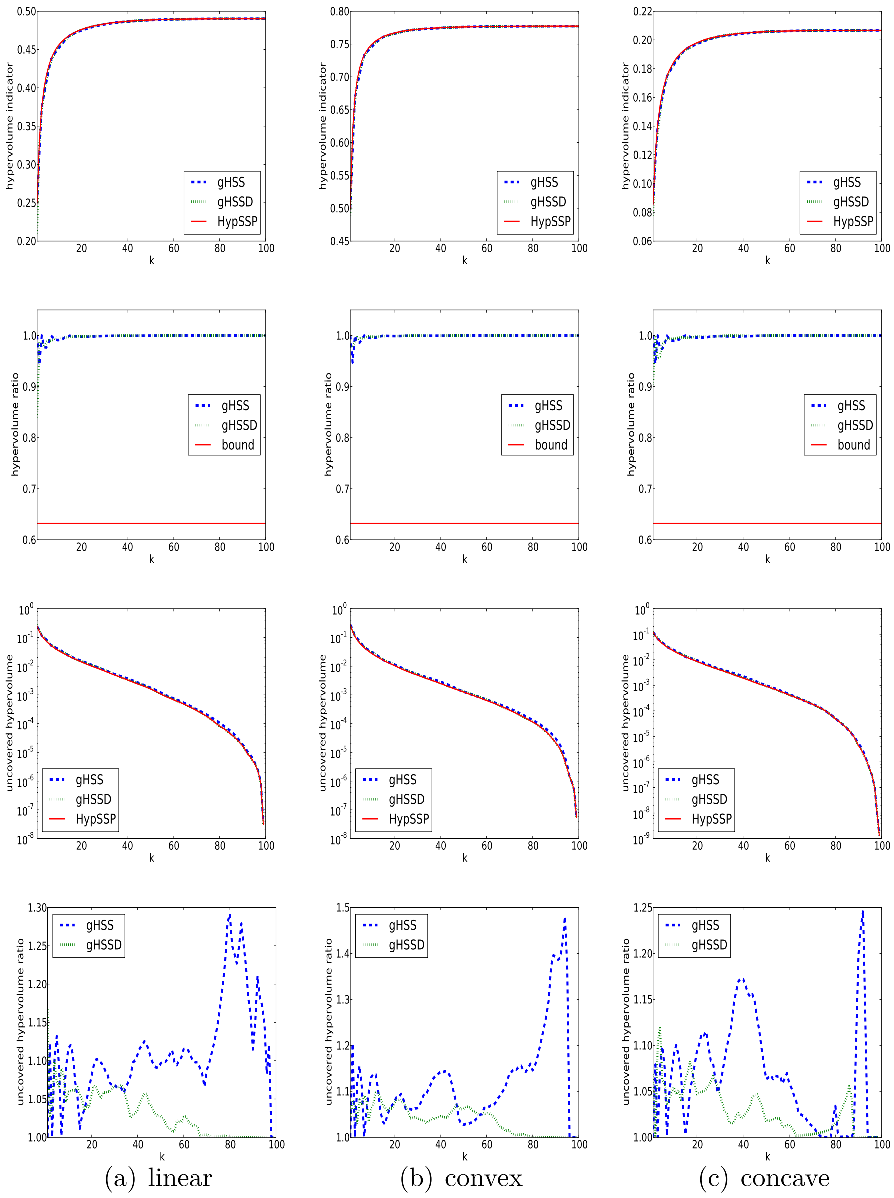
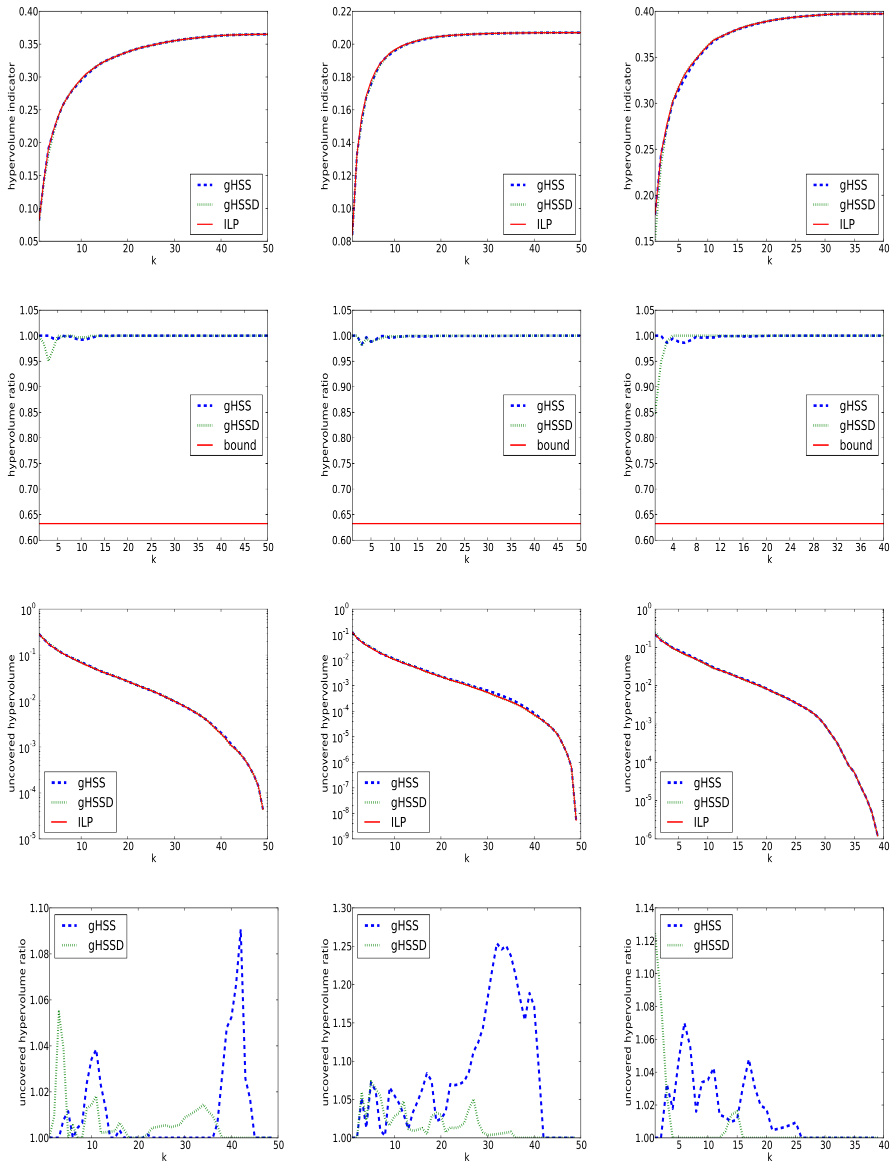


Figure 4.20: The hypervolume indicator (first row), the hypervolume ratio (second row), the uncovered hypervolume (third row) and the uncovered hypervolume ratio (forth row) for a fixed set size $n = 100$ and growing subset size k in a 2-dimensional problem.



(a) concave (b) degenerate (c) convex

Figure 4.21: The hypervolume indicator (first row), the hypervolume ratio (second row), the uncovered hypervolume (third row) and the uncovered hypervolume ratio (forth row) for a fixed set size $n = 50$ (concave and degenerate) and $n = 40$ (convex) and growing subset size k in a 3-dimensional problem.

	hypervolume ratio		uncovered hypervolume ratio					
	gHSS	gHSSD	gHSS			gHSSD		
data set	min	min	min	avg	max	min	avg	max
linear	**0.9361**	0.8394	1.0	1.1064	1.5322	1.0	**1.0323**	**1.1685**
convex	**0.9439**	0.8227	1.0	1.1020	2.9303	1.0	**1.0348**	**1.3783**
concave	**0.9351**	0.8376	1.0	1.0968	2.5803	1.0	**1.0310**	**1.1236**
wave-1	**0.9611**	0.8577	1.0	1.0985	2.7231	1.0	**1.0333**	**1.5172**
wave-2	**0.9837**	0.9243	1.0	1.1019	1.7586	1.0	**1.0324**	**1.3633**
wave-3	**0.9836**	0.9176	1.0	1.1001	**1.4233**	1.0	**1.0326**	1.4446
wave-4	**0.9813**	0.9658	1.0	1.1017	8.9046	1.0	**1.0331**	**1.2895**
wave-5	**0.9256**	0.8259	1.0	1.0945	2.6936	1.0	**1.0330**	**1.2827**
wave-10	0.8929	**0.9151**	1.0	1.0965	1.7817	1.0	**1.0310**	**1.2565**

Table 4.1: Minimum hypervolume ratio, and minimum, average and maximum uncovered hypervolume ratio of the approximations for the 2-dimensional HSSP for each data set, and considering all $k \in \{1, \ldots, n\}$, and all sizes n tested.

	hypervolume ratio		uncovered hypervolume ratio					
	gHSS	gHSSD	gHSS			gHSSD		
data set	min	min	min	avg	max	min	avg	max
linear	**0.9416**	0.8855	1.0	1.1937	2.3966	1.0	**1.0486**	**1.1574**
convex	**0.9698**	0.8475	1.0	1.0130	**1.0896**	1.0	**1.0073**	1.1247
concave	**0.9847**	0.9504	1.0	1.0156	1.1791	1.0	**1.0062**	**1.1545**
degenerate	0.9147	**0.9715**	1.0	1.0696	1.7081	1.0	**1.0196**	**1.1211**

Table 4.2: Minimum hypervolume ratio and minimum, average and maximum uncovered hypervolume ratio of the approximations for the 3-dimensional HSSP for each data set, and considering all $k \in \{1, \ldots, n\}$, and all sizes n tested.

close to 1. Finally, the tables show that the minimum hypervolume ratio achieved by gHSS in a data set is generally higher than those achieved by gHSSD.

Because the hypervolume of the optimal (and the greedy) subsets rapidly achieve values close to the hypervolume of the whole set, as k grows, the hypervolume ratio does not provide much information about which greedy algorithm provides better approximation. Therefore, the third and forth rows in Figures 4.20 and 4.21 provide more insight. The third row plots the uncovered hypervolume, which shows how much of the hypervolume of the full set is left out by the greedy and optimal subsets. The plots show how small the uncovered hypervolume is, which justifies the hypervolume ratios being close to one. For example, in Figure 4.20(b), for $k = n/2$, the hypervolume of the optimal subset is close to 0.775, and the corresponding uncovered hypervolume is on the order of 10^{-3}. Even if a greedy algorithm finds a subset that leaves out 10 times more area than the optimal subset, the approximation ratio is at least 0.988.

The forth row of plots in the figures show the uncovered hypervolume ratio while the right hand-side of Tables 4.1 and 4.2 summarize the results concerning all tests performed for each data set. The plots show that, in spite of gHSS providing, in general, better minimum value of hypervolume ratio (the higher the better) than gHSSD, the uncovered hypervolume ratio (the less the better) of gHSS seems slightly worst and less stable than of gHSSD, even for small values of k. The table results also confirm that, on average, gHSSD has better uncovered hypervolume ratio and, in most cases, also provides lower maximum values of this ratio than gHSS.

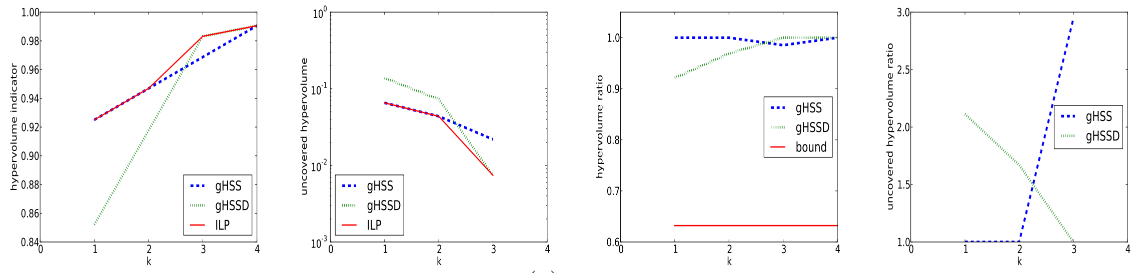
As there is a data set for gHSS and another for gHSSD for which the uncovered hypervolume ratio can grow arbitrarily, Figures 4.22 and 4.23 show how both algorithm behave in those two data sets. Figures 4.22 shows the hypervolume, the uncovered hypervolume and the two corresponding ratios, for the bad data set for gHSSD, which was described in Section 3.4.5 considering maximization (see in the description of gHSSD [D1]). Since minimization is assumed here, that data set was transformed by taking the negative of the points, adding 2 to every coordinate and normalizing all points by dividing each coordinate by 2, i.e., $X = \{(1-\epsilon, 1-\epsilon, 1-\epsilon)/2, (1-\epsilon+\sigma, 1, 1)/2, (1, 1-\epsilon-\sigma, 1)/2, (1, 1, 1-\epsilon-\sigma)/2\}$, where $\epsilon = \frac{1}{2\kappa d^2}$ and $\sigma = d^2\epsilon^2$. The reference point was set to $(1, 1, 1)$. Figure 4.22 shows the results for $n = 4$ and two settings of κ, namely, for $\kappa = 1$ and $\kappa = 100$. Figure 4.22 shows indeed that the uncovered hypervolume ratio is higher than κ for gHSSD and $k = 1$. However, the hypervolume ratio is higher than 0.92 and, as κ grows, the amount of uncovered hypervolume becomes a lot smaller than the hypervolume indicator of the subsets. Thus, despite the growth of the uncovered hypervolume ratio, the hypervolume ratio gets even closer to 1. gHSS shows a better performance in this particular data set than gHSSD, as expected.

Figures 4.23 is analogous to Figure 4.22 but is concerned with the bad data set for gHSS. The data set used is the one described in the proof of theorem 4.1, but normalized and translated to the positive orthant. The $d = 2$ case is considered and the reference point was set to $(1, 1)$. Figure 4.23 shows the results for $n = 3$ and two settings of ε, namely, for $\varepsilon = 1$ and $\varepsilon = 100$. The plots confirm that indeed, the uncovered hypervolume ratio may grow arbitrarily for gHSS as well. Once again, although the unconvered ratio visibly grows as ε grows, so does the difference between the hypervolume of the selected subset in comparison to the uncovered hypervolume. Consequently, in spite of the high uncovered hypervolume ratio, the hypervolume ratio is close to 1.

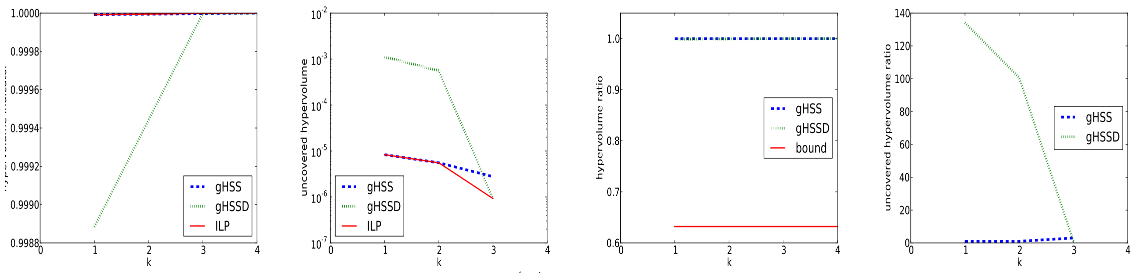
To conclude, both greedy algorithms find good approximations to the HSSP, in particular for $k > 2$, although gHSSD seems slightly more stable. Both approaches were shown to obtain consistently and considerably better approximation ratios to the HSSP (hypervolume ratio) than their theoretical bounds, even tough the obtained approximation ratio to the HSSPComplement (uncovered hypervolume ratio) can grow arbitrarily with any of the two approaches.

4.5.2 Selected Subsets

Figure 4.24 shows the subsets selected with the greedy algorithms, gHSS and gHSSD, and with the exact algorithm, HypSSP, given $k = 1, 2, 3, 4, 10$ and the 2-dimensional linear data set. The subsets selected by HypSSP (middle column) visibly approximate a uniformly distributed point set. This is expected as the optimal μ-distribution in a linear front is such that all points are equally spaced. This also means that, optimal subsets for k and $k + 1$ are not expected to have points in common (when $k < n/2$). Therefore, it is very likely that even if gHSS finds the optimal solution for k points, it will not be able to find the optimal solution for $k + 1$ points. For example, gHSS found the optimal solution for $k = 1$, but not for $k = 2$. However, the points selected by gHSS for $k = 2$ are in the optimal solution for $k = 3$ and

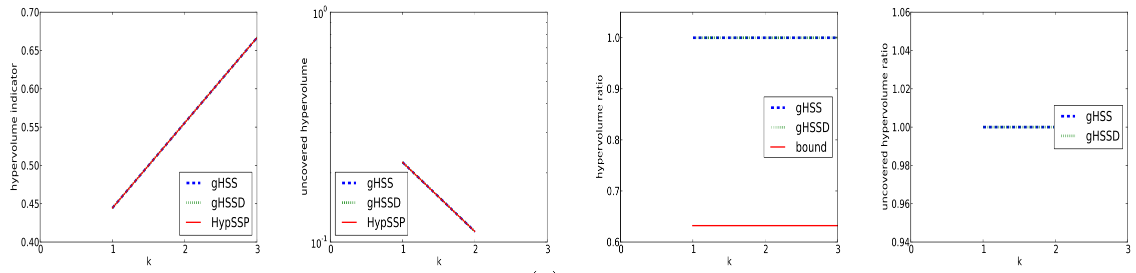


(a) $\kappa = 1$

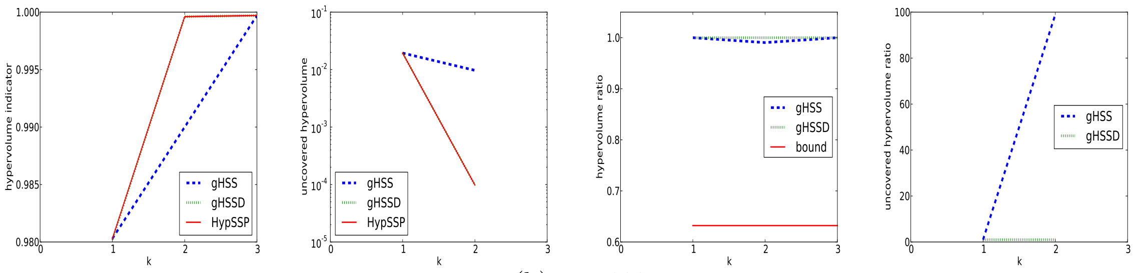


(b) $\kappa = 100$

Figure 4.22: The hypervolume indicator, the uncovered hypervolume and the corresponding ratios, respectively, from left to right, for the bad data set for gHSSD, in $d = 3$, with $n = 4$ and two different settings of κ.



(a) $\varepsilon = 1$



(b) $\varepsilon = 100$

Figure 4.23: The hypervolume indicator, the uncovered hypervolume and the corresponding ratios, respectively, from left to right, for the bad data set for gHSS, in $d = 2$, with $n = 3$ and two different settings of ε.

therefore, gHSS also finds the optimal solution for $k = 3$ but not for $k = 4$.

Comparing the subsets selected by gHSS and gHSSD, gHSSD clearly selected a worse subset for $k = 1$, and a slightly better one for $k = 2$ than gHSS did. For the remaining plotted cases they both found good subsets, i.e., with hypervolume values close to the optimal ones, and whose distributions become more similar to the optimal ones as k grows. Figure 4.25 shows the selected subsets for other fronts, where $n = 100$ and $k = 10$, and the conclusions are similar. Both greedy approaches were able to find more or less well distributed subsets. gHSSD found slightly better subsets for the convex and the concave data sets, while gHSS found slightly better subsets for the wave data sets regarding the respective hypervolume indicator values.

Similarly to Figure 4.24, Figure 4.26 shows the selected subsets for $k = 1, 2, 3, 4$ and 10, but for the $d = 3$ case on a convex front, and shows the corresponding volumes. Additionally, Figure 4.27 shows the volume left out by the discarded points (the uncovered hypervolume). Once more, gHSS and gHSSD sometimes find non-optimal subsets but these are usually well distributed and the corresponding hypervolume values are close to that of the optimal subset. Moreover, as k increases, the greedy subsets become more similar to the optimal one. Figures 4.28 shows more examples on other fronts, and the observations are similar. In this case, the hypervolume of the subsets chosen by gHSSD were slightly better than those chosen by gHSS.

Overall, the results suggest that both greedy approaches are able to find more or less well distributed subsets with hypervolumes close to the optimal ones, and, in general, none of them can be said to consistently find better subsets than the other.

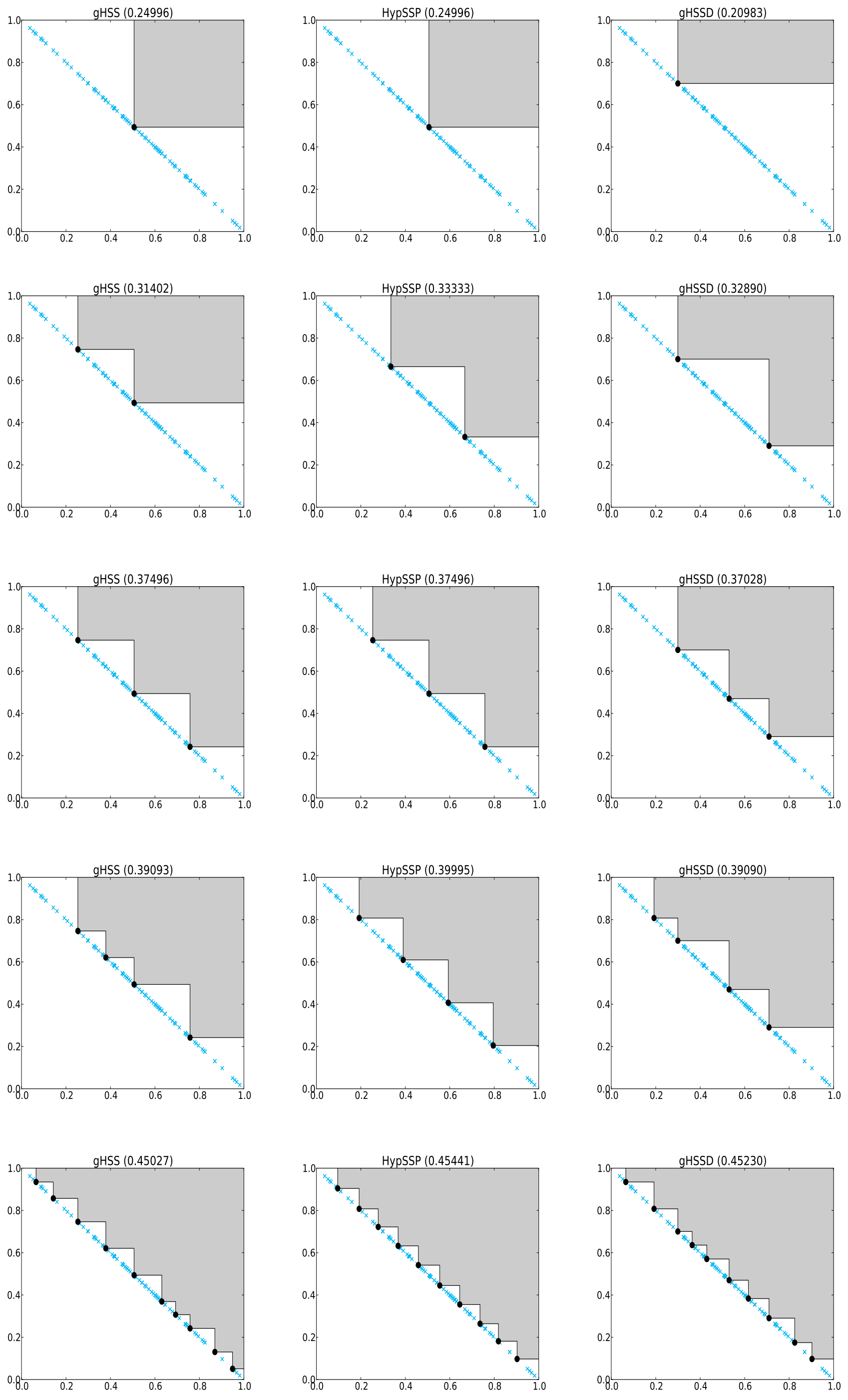


Figure 4.24: Subsets produced by gHSS, HypSSP and gHSSD, respectively, from left to right, for a 2-dimensional linear front. The selected solutions are represented as black dots, and the region dominated by them is shaded gray. The corresponding area is given in the title of each plot. The remaining solutions (blue crosses, $\times$) are discarded. The data corresponds to cases where $n = 100$ and rows from first to last represent, respectively, $k = 1, 2, 3, 4, 10$.

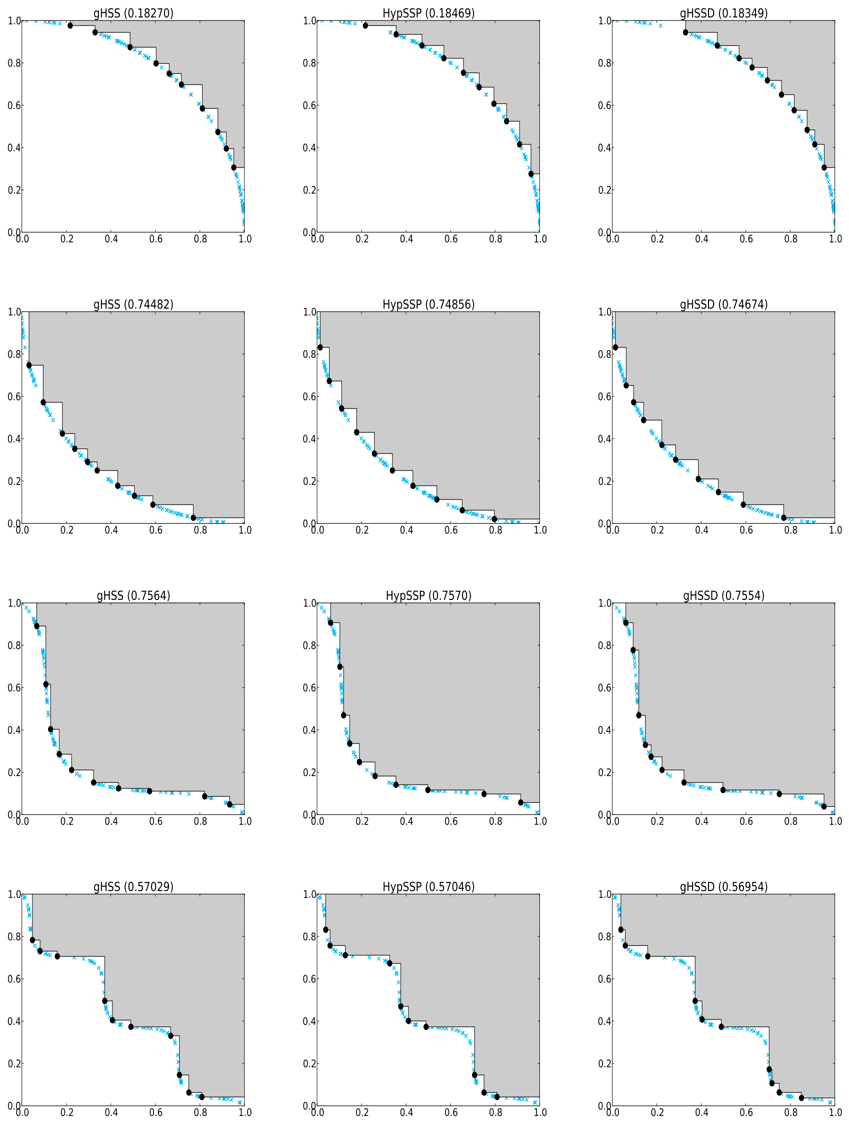


Figure 4.25: Subsets produced by gHSS, HypSSP and gHSSD respectively, from left to right, for 2-dimensional concave and convex fronts (top two rows), and for the wave fronts, i.e., mixed convex/concave fronts (bottom two rows). The selected solutions are represented as black dots, and the region dominated by them is shaded gray. The corresponding area is given in the title of each plot. The remaining solutions (blue crosses, $\times$) are discarded. The data corresponds to cases where $n = 100$ and $k = 10$.

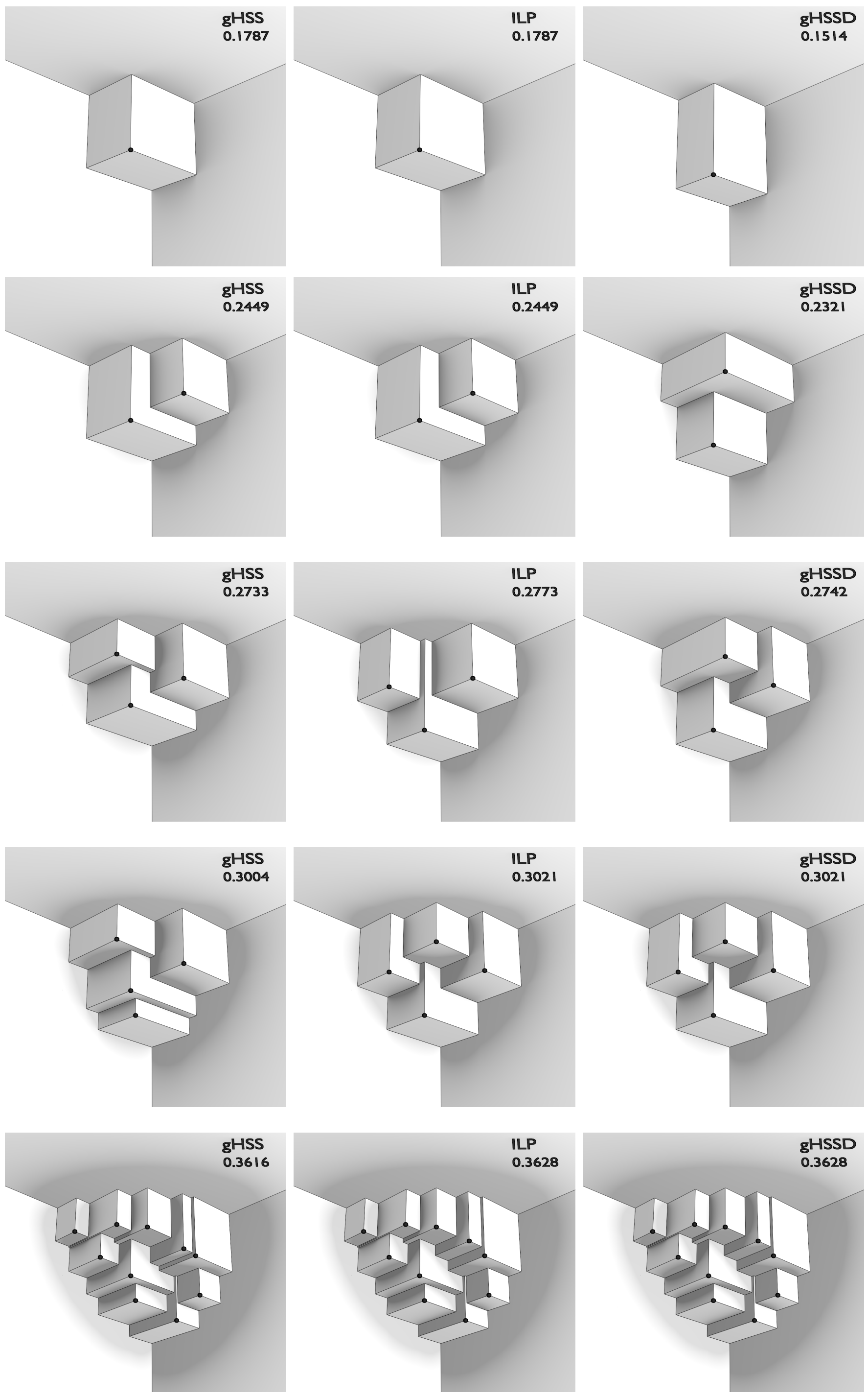


Figure 4.26: The subsets chosen (with solid volume) by each algorithm from a set of $n = 40$ points in a 3-dimensional convex front. The rows represent, from first to last, the solutions for $k = 1, 2, 3, 4, 10$. The hypervolume indicator for each subset is in the upper right corner as well as the name of the algorithm that produced it.

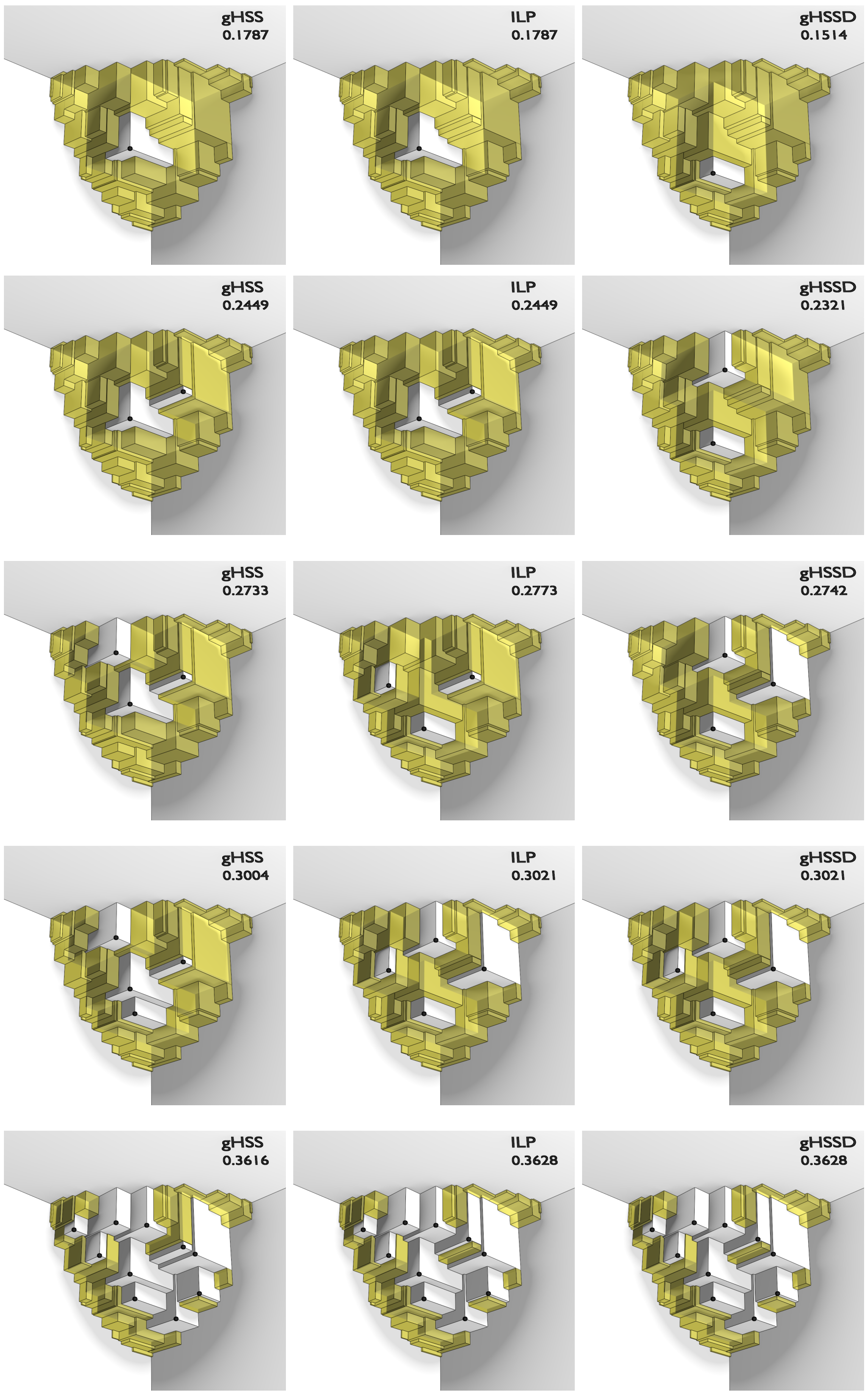


Figure 4.27: The same as Figure 4.26 but with the contribution of the discarded points shown in transparent yellow.

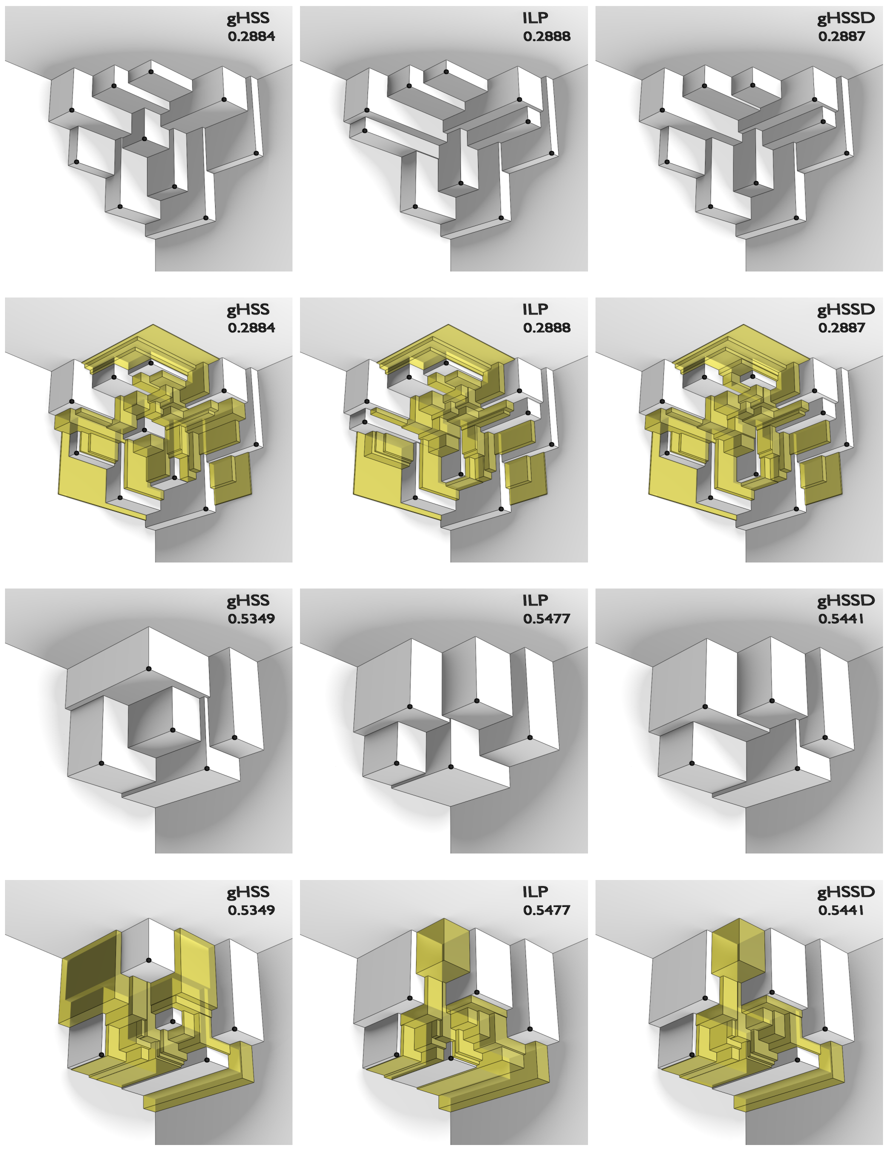
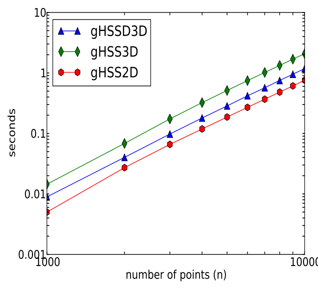


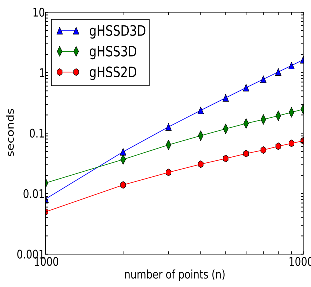
Figure 4.28: The subsets chosen by each algorithm in 3-dimensional fronts. The data in the first two rows correspond to the concave front, where $n = 40$ and $k = 10$. The top row shows the subsets chosen with solid volume, and the bottom row shows, in addition, the contribution of the discarded points with transparent yellow. Analogously, the bottom last rows correspond to the linear front, where $n = 20$ and $k = 5$. The hypervolume indicator for each subset is in the upper right corner as well as the name of the algorithm that produced it.

4.5.3 Runtime

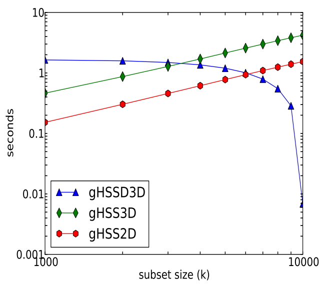
Figure 4.29 and 4.30 shows the behaviour of gHSS2D, gHSS3D and gHSSD3D (using HVC3D-U) for different settings of n and k. The average runtimes of 13 runs on the convex and degenerate fronts (the latter only for the algorithms for the $d = 3$ case) are depicted (other fronts produced similar results to those on the convex data set). Figures 4.29(a) and 4.29(d) clearly show the quadratic growth of the runtime of all algorithms for growing set size n and $k = n/2$. Still, selecting $k = 5000$ out of $n = 10000$ points on the convex data set was performed in less than 1 second in 2 dimensions, and, at most, in 2 seconds in the 3-dimensional case, where gHSSD3D took about 1 second and gHSS3D took about 2 seconds. On the degenerate data set, the performance of gHSSD3D was similar, but gHSS3D performed worse, where selecting half of $n = 10000$ required about 5 seconds.



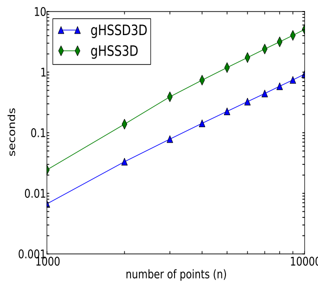
(a) convex, $k = n/2$



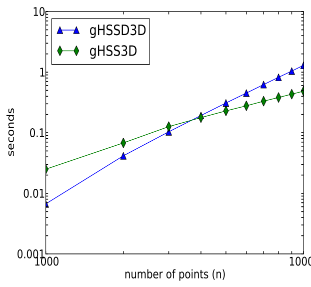
(b) convex, $k = 500$



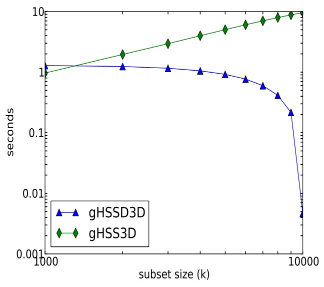
(c) convex, $n = 10^5$



(d) degenerate, $k = n/2$



(e) degenerate, $k = 500$

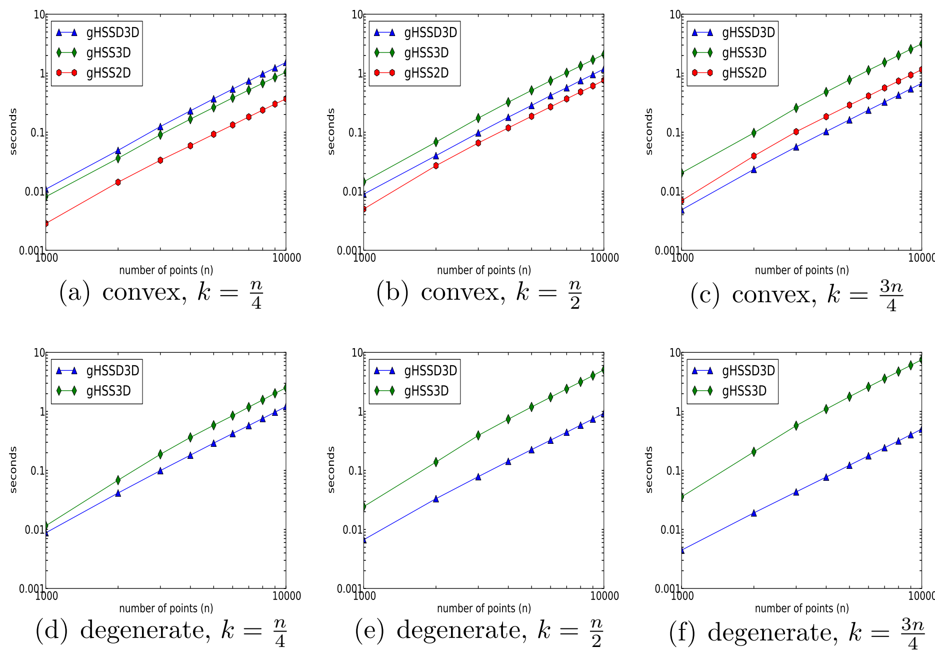


(f) degenerate, $n = 10^5$

Figure 4.29: Runtime of gHSS2D, gHSS3D and gHSSD3D for the convex (first row) and the degenerate (second row) data sets for: varying n and k (first column); varying n and a fixed k (second column); and a fixed n and varying k (third column).

The second column of plots (Figures 4.29(b) and 4.29(e)) shows the runtime for growing set size n and fixed $k = 500$. Regarding gHSS2D and gHSS3D, although runtimes appear to grow slightly faster than linearly for small n, for $n > 3000$, the runtime growth of both algorithm is essentially linear. The same can be observed for growing subset size k and fixed $n = 10000$, in the third column of plots (Figures 4.29(c) and 4.29(f)). The observed runtimes are in agreement with their time complexity of $O(n(k + \log n))$. Regarding gHSSD3D, its runtime growth when k is fixed, in the second column of plots, is essentially quadratic. When n is fixed, in the third column of plots, contrary to the other algorithms and the other settings of n and k, the runtime of gHSSD3D decreases as k grows. The behavior of gHSSD3D in these plots is in agreement with its time-complexity of $O(n(n-k+\log n))$, which has a quadratic growth of the runtime with increasing n, and a decreasing runtime with growing k and fixed n.

Seeing gHSS be faster for smaller values of k and gHSSD be faster for higher values of k was expected due to each one's incremental/decremental nature. Figures 4.29 and 4.30 set a threshold on the value of k with respect to n, from which one algorithm is faster than the other, in the $d = 3$ case. Namely, gHSSD3D is faster for $k > \frac{n}{3}$ and $k > \frac{n}{8}$ for the convex and the degenerate data set, respectively. The reason why the threshold is not always very close to $\frac{n}{2}$ as could be expected, may be explained by the additional overhead gHSS3D has in comparison to gHSSD3D. Firstly, regarding the update procedure, gHSS3D has to repeat it three times considering a different objective order each time, while gHSSD3D only needs to sweep points once. Secondly, in the decremental case (gHSSD3D), a point is discarded at each iteration and once that happens, that point is not needed anymore, and thus is not visited again, while in the incremental case (gHSS3D), all points have to be visited everytime a new point is selected, because every point is either not selected yet and its contribution has to be updated, or it is a selected point and it may be needed to delimit the contributions of the other points. Therefore, gHSS3D works with a constant number of points in every iteration of the algorithms main loop, while gHSSD3D works with a decreasing number of points.



(a) convex, $k = \frac{n}{4}$ (b) convex, $k = \frac{n}{2}$ (c) convex, $k = \frac{3n}{4}$

(d) degenerate, $k = \frac{n}{4}$ (e) degenerate, $k = \frac{n}{2}$ (f) degenerate, $k = \frac{3n}{4}$

Figure 4.30: The runtime of gHSS2D, gHSS3D and gHSSD3D for the convex (first row) and the degenerate (second row) data set. Each column represents a different setting of k. From left to right, k is $\frac{n}{4}$, $\frac{n}{2}$ and $\frac{3n}{4}$, respectively.

Figure 4.30 shows the performance of gHSS2D, gHSS3D and gHSSD3D for growing n and k, where k is set to either $\frac{n}{4}$, $\frac{n}{2}$ or $\frac{3n}{4}$. These plots shows that the smaller the ratio $\frac{n}{k}$ is, the faster gHSSD3D is in comparison to gHSS3D. As observed previously, gHSS3D only outperforms gHSSD3D on the convex front and with small k, in this case, for $k = \frac{n}{4}$. In the remaining cases, gHSSD3D runs up to 20 times faster than gHSS3D, and is faster even when compared to gHSS2D on the convex data set, when $k = \frac{3n}{4}$.

The very small amount of time spent by the greedy algorithms with n set to a few hundred of solutions is very appealing for environmental selection in EMOAs. In a typical scenario such as a population of $\mu = 200$ individuals and $\lambda = \mu$ offspring, each environmental selection step should not require more than 10^{-2} seconds with either gHSS3D or gHSSD3D, as can be inferred from Figures 4.30(b) and 4.30(e). The runtime is expected to be even smaller because it is not likely that all $\mu + \lambda$ individuals are consistently nondominated.

To conclude, both gHSS3D and gHSSD3D are very fast in practice, especially when compared to the amount of time that an algorithm to compute HSSP exactly would require. Depending on the particular setting of n and k, one can be more advantageous than the other with respect to runtime. For growing n and fixed k, gHSSD3D shows a quadratic behavior while gHSS shows almost linear behavior and therefore, gHSS3D is more advantageous than gHSSD3D in such scenario. Otherwise, gHSSD3D is preferable to gHSS3D unless k is very small compared to n.

4.5.4 Remarks on the Greedy Algorithms

In Section 4.3, the decremental greedy algorithm was shown to provide a (k/n)-approximation to the optimal solution of the HSSP, and new instances for $d = 3, 4$ were proposed. The new algorithms gHSSD3D and gHSSD4D have a worst-case time complexity of $O(n(n-k) + n \log n)$ and $O(n^2(n-k))$, respectively, and both improve the state-of-the-art algorithms by a $\log n$ factor. In Section 4.4, incremental greedy algorithms for the HSSP in two (gHSS2D) and three (gHSS3D) dimensions were proposed, providing a $(1 - 1/e)$-approximation to the optimal solution in each case. Both of them have a worst-case time complexity of $O(n(k + \log n))$, which is better than other incremental greedy instances based on iterating over existing algorithms to compute the OneContribution problem (see Section 3.4.5).

The two-dimensional version of the incremental greedy algorithm, gHSS2D, has a worst-case time complexity similar to the complexity of exact algorithms for the same problem [37, 100]. However, it does have the advantage of being very simple to implement and very fast in practice. The time complexity of the three-dimensional version, on the other hand, and of gHSSD3D as well, is considerably lower than that of the corresponding exact algorithms, which are $n^{O(\sqrt{k})}$ (see [C6]), and $\Omega(n^3)$ due to the $\Omega(n^3)$ constraints in the ILP formulation (see [C4]). Moreover, the greedy algorithms are more memory efficient, requiring only $O(n)$ space. In the four-dimensional case, the difference in the time complexities is even more evident when comparing that of gHSSD4D with the $O(n^2 \log n + n^{n-k})$ time complexity of the only known exact algorithm (see [C1]).

Regarding the quality of the approximation, the results obtained experimentally with gHSS (and even with gHSSD) were much better than the guaranteed approximation factor, staying within at least 0.89 of the optimal values, in comparison to the theoretical $(1 - 1/e) \simeq 0.63212$. Although the approximation guarantee of k/n of the decremental counterpart is poor for small values of k, the experimental results suggest that this ratio is far from being tight as the observed ratio stayed within at least 0.82. Despite the known approximation guarantees, both approaches provided good approximations. Regarding runtime, the former was shown to be more advantageous for smaller values of k and the latter for greater values of k.

The theoretical and experimental results show that the new algorithms, for both approaches, should provide an interesting alternative to much more computationally expensive exact algorithms for hypervolume-based multiobjective selection and/or archiving in the three-objective case. Moreover, the proposed algorithms are suited for post-processing evaluation of EMOAs (fixed k and variable n scenario), particularly the incremental greedy one, providing bounds on the hypervolume of the best subset of k solutions among all solutions generated by the EMOA up to given points in its execution. In this case, such bounds could be updated in $O((n+m)k+m \log m)$ when m new solutions are added to an existing set of n previously processed solutions.

From an algorithmic perspective, the algorithms for the three dimensional case are the most relevant as they can avoid recomputing everything from scratch. It is important to highlight that the usefulness of the update procedures in gHSS3D and gHSSD3D (i.e., the HVC3D$^+$) is not restricted to the incremental/decremental greedy approaches. Those update procedures are independent of the order in which points are selected/discarded, implying that they can be used in other scenarios, and can even be combined and used to add and to discard points arbitrarily, which could be useful for other types of greedy algorithms, for example.

4.6 Greedy Hypervolume-based Archiving Algorithms

Recall from Chapter 3.3 that a $(\mu + \lambda)$ Hypervolume-based archiving algorithm (or $(\mu + \lambda)$ Hypervolume-based archiver) purpose is to be consecutively used in an EMOA to perform environmental selection. It is used to select the μ solutions of the next population given the solutions in the μ-sized parent population and the λ-sized offspring set, and seeks the maximization of the hypervolume indicator of the selected μ-sized subset. An exact algorithm for the HSSP used for this purpose (given $k = \mu$) is classified as a *locally optimal archiving algorithm.* A decremental greedy subset-selection algorithm (gHSSD) by itself, is classified as a *(decremental) greedy archiving algorithm.* Analogously, an incremental greedy subset-selection algorithm (gHSS), by itself, is here classified as an *incremental greedy archiving algorithm.* The latter two, in general, do not guarantee that the hypervolume of the new population is better than that of the parent population. However, greedy algorithms can also be used in combination with the broader class of *nondecreasing archiving algorithms*, which provide such guarantee. A nondecreasing archiving algorithm (see Algorithm 4.10) uses an auxiliary algorithm to select a subset of μ solutions (see line 1), such as a greedy algorithm, and only accepts that subset if its hypervolume indicator is not lower than that of the parent population. Let us call an instance of Algorithm 4.10 where gHSSD is used in line 1 as *nondecreasing decremental greedy archiving algorithm*, and as *nondecreasing incremental greedy archiving algorithm*, if gHSS is used instead.

The studies on effectiveness and competitiveness allow to learn the limitations and guarantees that an archiving algorithm (or archiver, for short) provides to an EMOA that uses it for (environmental) selection, despite the variation operators it may use. These studies can show whether an archiving algorithm stands any chance of ending up finding an optimal μ-sized subset for the HSSP, whatever is the ini-

Algorithm 4.10 General nondecreasing archiving algorithm

```
Require: P // Population with size $\mu$
Require: Q // Offspring set with size $\lambda$
1: $P' \leftarrow \mathsf{subsetSelection}(P \cup Q, \mu)$
2: if $H(P') \geq H(P)$ then
3:     $P \leftarrow P'$
4: return P
```

tial population. If not, they may be able to state how close it can get (study of effectiveness). They can also show whether an unlucky sequence of offspring may dictate the loss of important points, and how adverse can that be (study of competitiveness). If the archiver's guarantees and limitations are known, then, regardless of the variation operators chosen, it is also known that the algorithms are expected to perform better/worse than certain established bounds.

Note that the theoretical studies on effectiveness and competitiveness do not completely capture the dynamics of an Hypervolume-based EMOA as it is subject to variation operators that typically delimit the neighborhood of the current population, and consequently, the offspring population. Moreover, since these theoretical studies cover any possible problem instance, the bounds may not be tight for many of them. For example, in effectiveness studies, having a single initial population for which there is not a sequence of offspring such that the archiving algorithm is able to reach the optimal solution, is enough to say that it is not effective. However, this may be just a particular initial population among many for which there is such a sequence. On the other hand, the existence of a sequence for which the archiving algorithm is effective, does not ensure that, in practice, it can be generated, as the generation of sequences will be restricted by variation operators. Competitive analysis can be more informative in the sense that it may allow us to say that, whichever the initial population and offspring sequence (and thus, variation operators), then a certain quality guarantee is always achieved. Despite the (possibly) limited conclusions that may be drawn from these theoretical studies regarding effectiveness and competitiveness, they are an important first step to understanding archiving algorithms.

In this Section, the greedy and exact archiving algorithms are tested on different fronts and different settings of μ and λ and, when possible, the experimental results are compared to the known theoretical ones.

4.6.1 Convergence Quality

In the study by Knowles *et al.* [98], the $(\mu+1)$-locally optimal hypervolume-based archiving algorithm was studied by observing the hypervolume of the last archive for 30 permutations of a sequence of m mutually nondominated points. That is, the sequence of m points is used to simulate the environmental selection, where the first μ points compose the initial population, and the remaining ones compose a sequence of λ-sized chunks, which represent a sequence of offspring sets. Repeatedly, the archiving algorithm takes the current population and the new offspring set, and selects μ points to compose the new population. That study was restricted to small sequences of, at most, $m = 200$ points and μ was between 10% and 90% of m. With the current state-of-the-art algorithms, this study can now be extended for $\lambda > 1$, for bigger sequences and populations, and include a comparison to the optimal μ-sized subset to the HSSP given the m points (in the $d = 2$ case). In particular, the case of $\lambda = \mu$ for which the locally optimal archiver is effective can be tested. Moreover, the greedy and the locally optimal archiving algorithms can be compared.

A similar study to that in Knowles *et al.* [98] is provided here for the archiving algorithms based on the incremental and decremental greedy subset-selection algorithms, gHSS and gHSSD, respectively. When possible, these are compared against a (locally optimal) archiving algorithm using an exact algorithm for the HSSP. The intention of this study is to get an insight into how well the greedy archivers are able to keep the best solutions compared to one another and to a locally optimal archiver for different settings of μ and λ.

Note that, in theory, no nondecreasing $(\mu+\lambda)$-archiver is expected to consistently retain the best subset among all solutions seen (see Section 3.3.2). Moreover, only the increasing archivers (which include locally optimal archivers) such that $\lambda \geq \mu$, are effective, i.e., there is at least a sequence of offspring that allows the archiver to achieve the global optimum subset. The following study also allow us to observe how close or detached the theory is from the average performance of the archivers. For example, to check if known bounds are, in general, tight, or whether they are too attached to a particular sequence and that in practice the algorithm consistently performs better/worse.

Experimental Setup

Five types of archiving algorithms were tested: a locally optimal, an incremental greedy, a nondecreasing incremental greedy, a decremental greedy, and a nondecreasing decremental greedy archiving algorithm. In the plots, the archiving algorithms will be identified by the name of the subset-selection algorithm used, i.e., the locally optimal one is identified by HypSSP (only for the $d = 2$ case), and the incremental and decremental greedy ones are identified by gHSS and gHSSD, respectively. For the nondecreasing version of the greedy archiving algorithms, the algorithm used will be proceeded by "-ND", i.e., gHSS-ND and gHSSD-ND.

Several data sets (see Section 3.5) for the $d = 2$ and $d = 3$ cases were used, and for each one, a sequence of $m = 10^5$ (nondominated) points was randomly generated. For each data set, 30 permutations of the original sequence were considered. Every combination of an archiver and a parameter setting (of μ and λ) was tested on the same 30 sequence permutations. Therefore, any two archivers using the same setting for μ and λ, visit points exactly in the same sequence of chunks of points. The parameter setting considered were: $\mu \in \{10, 100, 200, 500\}$ and $\lambda \in \{1, \lfloor\frac{\mu}{4}\rfloor, \frac{\mu}{2}, \mu, 2\mu, 4\mu, 10\mu\}$. For $d = 3$, running ILP just for $m = 50$ would take hours and because preliminary experiments showed that results for small values of m, μ and λ did not provide additional relevant information, only the results for the greedy archivers are presented in that case.

Plots

Figures 4.31 and 4.32 show the results in the $d = 2$ case for the concave and the wave-2 data sets, respectively. Each plot refers to an archiving algorithm, and shows the boxplots for the hypervolume ratio between the hypervolume indicator of the final archive, and the optimal μ-sized subset for the HSSP given the set of all m

points. The exception is the boxplots marked with "*" in the x-axis, which refer to the hypervolume ratio of the subset selected by the subset-selection algorithm used within the archiving algorithm, given the set of all m points. Figure 4.33 shows the results for the convex data set in the $d = 3$ case. In this case, with the absence of the optimal subsets, only the results for the greedy algorithms are shown and the y-axis represents the hypervolume indicator and not the hypervolume ratio. However, recall that gHSSD is exact for $\lambda = 1$ and so it can somehow be used as reference.

Exact subset selection: For each value of μ, the locally optimal archiver (see HypSSP) performed better than all greedy archivers, independently of their λ setting. This is observed both for the average ratio which is higher with the locally optimal archiver, and for the ratio variation which is smaller than for most greedy archivers. As expected in this case, the larger is λ (with fixed μ), the more accurate the archiver is.

Increasing μ: As μ increases, all archiving algorithms became more accurate, i.e., achieve better hypervolume ratios and vary less (note that, in spite of the box size in the figures, from the top row to the bottom row, as μ grows, the range of the y-axis gets smaller). For example, in Figure 4.31, for μ equal to 10, 100, 200, and 500, all algorithms achieve a ratio higher than 0.9860, 0.9986, 0.9993, and 0.9998, respectively.

Increasing λ: In general, greater values of λ were only (slightly) better for the locally optimal archiver (see HypSSP). For the two greedy decremental archivers (see gHSSD and gHSSD-ND), the results were better for smaller values of λ. The same is observed for the nondecreasing incremental greedy archiver (see gHSS-ND) in the $d = 2$ case, whereas in the $d = 3$ case, in most settings of μ the approximation quality achieved decreases as λ increases up to $2 \cdot \mu$, and improves as λ grows from that value up to $10 \cdot \mu$. The incremental greedy archiver (see gHSS), is the most consistent archiver of all, as changing λ barely affects the quality of the final populations. In most cases, the most advantageous setting of λ for the greedy archivers was $\lambda = 1$, even with the incremental greedy ones (see gHSS). This observation was expected for the decremental greedy archivers (see gHSSD).

Nondecreasing vs (possibly) decreasing: In most cases, the nondecreasing version of greedy archivers (see gHSS-ND and gHSSD-ND) performed better for the same settings of μ and λ. The nondecreasing version (see gHSS-ND) of the incremental greedy archiver (see gHSS) is not as consistent across different settings of λ as the later, particularly in the $d = 2$ case.

Incremental greedy vs decremental greedy: The known approximation ratios for gHSS and gHSSD, of $(1 - 1/e)$ and k/n, respectively, provide a better guarantee by gHSS for greater values of λ (i.e., small k), and a better guarantee by gHSSD for smaller values of λ. Despite this, in most cases the decremental greedy archiver (see gHSSD) produced better results than the incremental one (see gHSS) for the same setting of μ and λ. This is also observed for the nondecreasing versions. Among all, the nondecreasing decremental greedy archive (see gHSSD-ND) was the one that more consistently produced better results than the other greedy archivers, particularly with $\lambda = 1$.

Final greedy archive vs overall greedy solution: An interesting results is that, comparing to the overall greedy solution of gHSSD given the set of m points (labeled with * in the plots), the final populations of the two decremental greedy archives (see gHSSD and gHSSD-ND) were better in most cases. Similar observations can be made for the greedy solution of gHSS when compared to the nondecreasing incremental greedy archiver (see gHSS-ND). This indicates that these greedy algorithms tend to find better subsets if they update the greedy solution as they receive point in sequences of small chunks (particularly if $\lambda < \mu$), than if the whole point set is known in advance.

Relating with theory: The result on competitiveness sets a lower bound that can be directly observed/confirmed in the experiments performed. The locally optimal (see HypSSP) archiver is μ-competitive. Therefore, it was known beforehand that the hypervolume ratio achieved by this archiving algorithm in the experiments was, at least, $\frac{1}{\mu}$. For example, for $\mu = 10$ and $\mu = 500$, the minimum hypervolume ratio guaranteed is 0.1 and 0.002, respectively. This is confirmed and is extremely far from the hypervolume ratios observed which were higher than 0.99 in all cases. Moreover, the hypervolume ratios observed for the nondecreasing greedy archivers were higher than 0.96 in all cases, even though there is no $(1.1338 - 0.1338/\mu - \epsilon)$-competitive nondecreasing archiver, which means that, for large μ, there is at least one case for which the nondecreasing archiver cannot achieve a ratio of 0.75.

The results presented show that there are slight differences in using an exact algorithm for the HSSP, or an incremental/decremental greedy algorithm to approximate the HSSP, within archiving algorithms. Although using an exact algorithm clearly outperforms the use of a greedy one, they all achieve absolute ratios values very close to 1, especially as μ grows. Nevertheless, the results indicate that, an exact algorithm for the HSSP should always be preferred (when an efficient one is available) to a greedy one independently of the setting of λ to be used, although one with $\lambda \geq \mu$ (for which the archiver is effective) is preferable. In particular, the experiments indicate that using an exact algorithm in a steady-state setting ($\lambda = 1$), such as in SMS-EMOA, should be preferable to using a greedy algorithm with any setting of λ, including in a generational ($\lambda = \mu$) setting. For $\lambda > 1$, and in the absence of an efficient exact algorithm, greedy algorithms are a good alternative, especially within a nondecreasing version of the archiver. In particular, the nondecreasing decremental greedy archiving algorithm showed slightly better results, especially for $\lambda \leq \mu$, where the smaller λ is, the better.

Note that the above conclusions/recommendations are based on the observed capability of the archiving algorithms to retain good solutions given the same initial population and sequence of offspring sets. However, these experiments do not cover the dynamics of EMOAs introduced, for example, by the variation operators. In combination with the archiver, these will influence the solutions that will be generated, and EMOA convergence. Consequently, EMOAs using different archiving algorithms will very likely generate different solutions, and in different sequences, even if the initial population, μ, and λ are the same. Moreover, it will not always

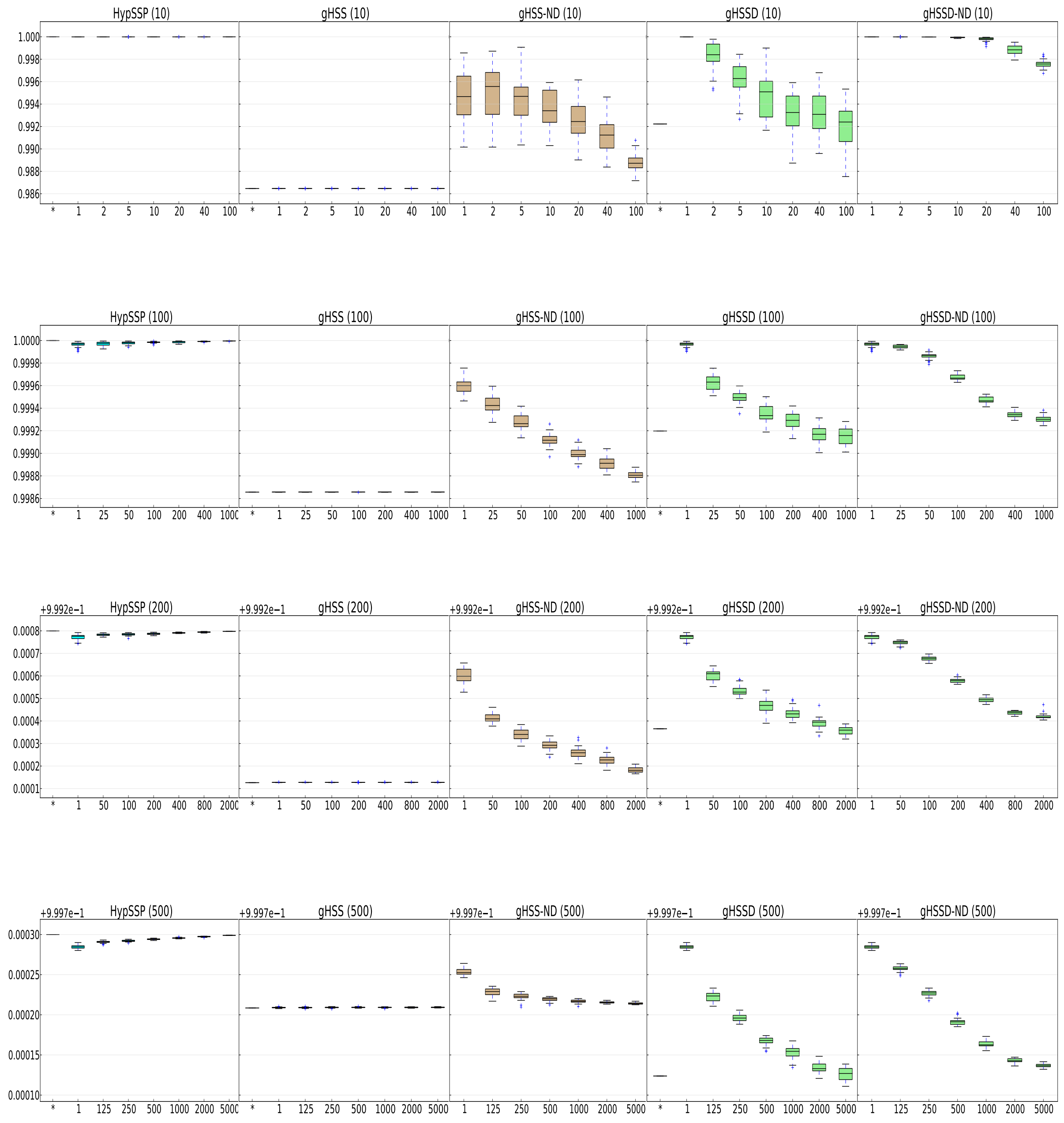


Figure 4.31: Results for 30 permutation sequences of $m = 10^5$ points on the concave data set in the $d = 2$ case. The title in each plot indicates the name of the subset-selection algorithm and, in parenthesis, the value of μ used (a different one per each row). The y-axis indicates the hypervolume ratio between the hypervolume indicator of the final archive and the optimal μ-sized subset for the HSSP given the m points. The x-axis indicates the values of λ used or an asterisk (“*”) referring to the μ-sized subset selected by the respective algorithm given the set of all m points.

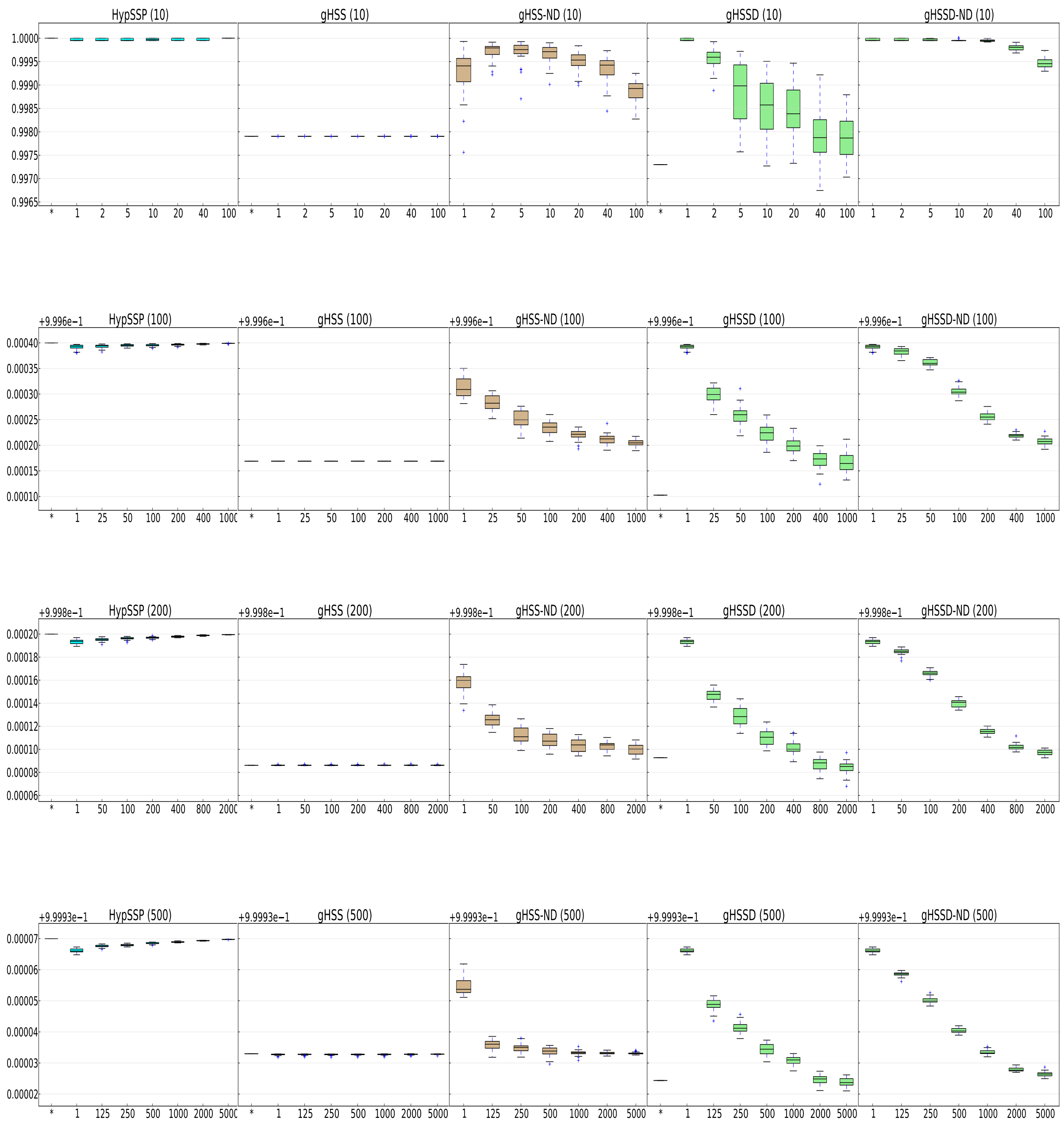


Figure 4.32: These boxplots are analogous to those in Figure 4.31 but with respect to the wave-2 data set in the $d = 2$ case.

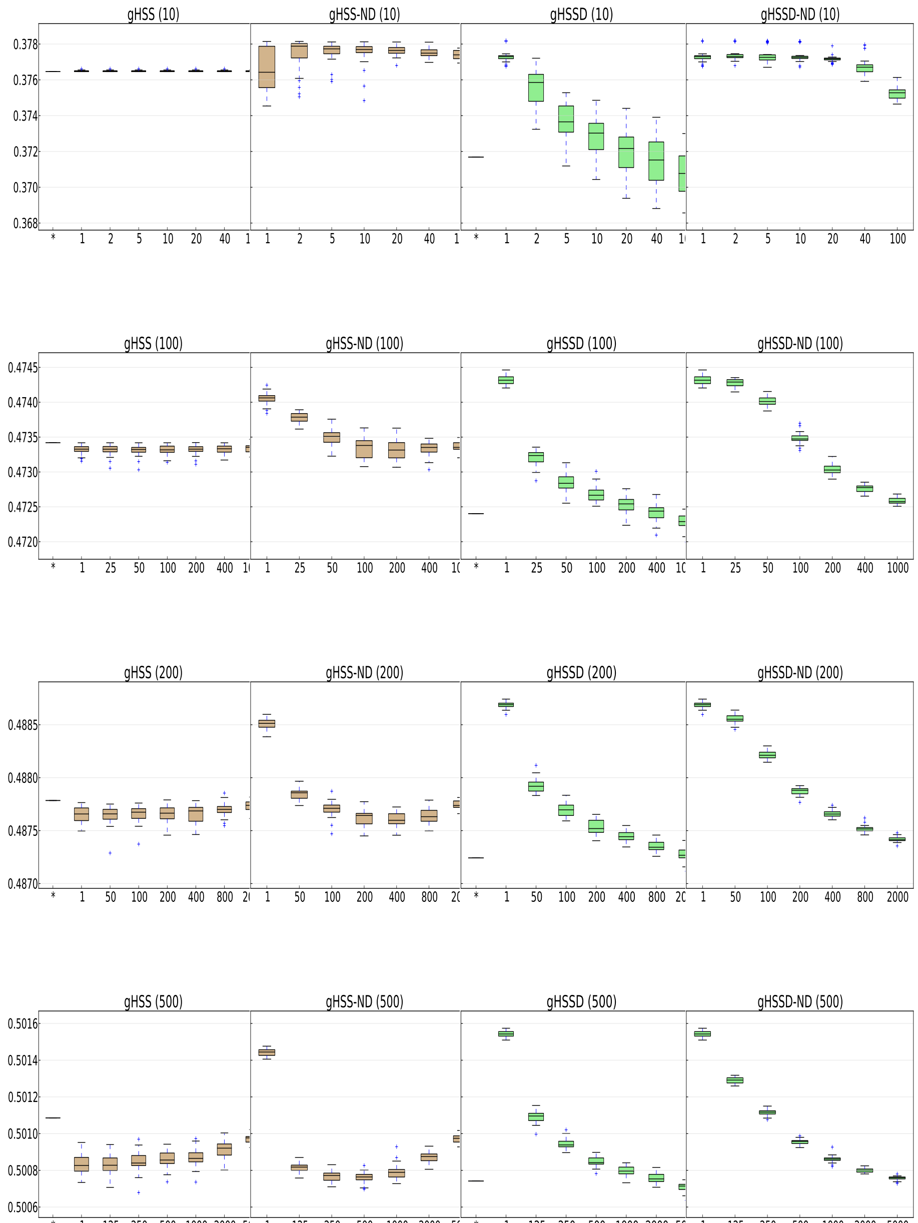


Figure 4.33: These boxplots are similar to those in Figure 4.31 but refer to the convex data set in the $d = 3$ case, and the y-axis indicate the hypervolume indicator instead of the hypervolume ratio.

be the case that the union of the μ individuals and the λ offspring will all be non-dominated, particularly in the earlier stages of the evolution. Therefore, there will be less than $\mu + \lambda$ solutions to choose from.

Finally, the experiments indicate that the theoretical results, although important for understanding the archivers under extreme situations, may be very far from most of practical cases.

4.7 Concluding Remarks

Computational problems related to the hypervolume indicator frequently arise in connection with the design, implementation, and experimental evaluation of evolutionary algorithms and other metaheuristics for multiobjective optimization. Arguably, the development of algorithms for such problems in the literature has taken three main directions to date, one aiming for algorithms that are fast in practice, especially for large numbers of objectives, a second one focusing on algorithm complexity in relation to the number of objectives, and a third directed at low-dimensional cases. The last direction typically encompasses two and three objectives, with occasional incursions into four objectives, which remain the most common use cases in multiobjective optimization in spite of growing interest in so-called many-objective optimization, and for which it has been possible to develop algorithms that are both asymptotically efficient, or even optimal, and very fast in practice.

In this chapter, new algorithms for the computation and update of hypervolume contributions were developed by building upon existing algorithmic approaches to the computation of the hypervolume indicator in three and four dimensions. A novel $O(n \log n)$-time preprocessing step for the three-dimensional case was the key ingredient in the development of $O(n)$-time algorithms for the subsequent computation of Hypervolume, OneContribution and AllContributions, as well as for the corresponding UpdateHypervolume and UpdateAllContributions problems in three dimensions, even under reference point changes. As a direct result, a novel algorithm for AllContributions in four dimensions was obtained, and a new time complexity upper bound of $O(n^2)$ was established for this problem. Using the proposed algorithms, the decremental greedy approximation to the HSSP (and an exact solution to the HSSP in the $k = n - 1$ case) can now be computed in $O(n(n-k) + n \log n)$ and $O(n^2(n-k))$ time, in three and four dimensions, respectively. The experimental results obtained indicate that the better complexity bounds achieved by the proposed algorithms do translate into considerable speed-ups in practice. Following the same principle of developing efficient algorithms aimed at low dimensions, $O(n(k + \log n))$ time algorithms for the incremental greedy approximation to the HSSP in the $d = 2, 3$ cases were proposed.

Although neither the incremental nor the decremental greedy approaches provide an approximation ratio to the HSSPComplement, they do provide an approximation ratio to the HSSP of $(1 - 1/e)$ and k/n, respectively, and in the experiments they performed much better than these approximation ratios. EMOA simulations where these greedy algorithms where used within archiving algorithms for environmental selection also showed promising results. These observations and the runtime performance of the new greedy algorithms indicate that these approaches may be good alternatives to the very expensive exact algorithms for the HSSP, and may be adequate for environmental selection in EMOAs. In fact, since the decremental greedy approach is optimal for $k = n - 1$, the new algorithms provide a speed up on the exact computation of HSSP in this case. The steady-state SMS-EMOA directly benefits from these speed-ups, and in $d = 3$ can benefit even more by taking advantage of HVC3D$^+$ to efficiently update the contributions of the individuals.

The usefulness of the proposed algorithms goes beyond decremental/incremental greedy selection and beyond environmental selection. For example, with efficient algorithms to compute AllContributions, these can be used for fitness assignment. Another example is the fact that the developed algorithms for updating contributions have already been useful in the proposal of the first branch-and-bound algorithm for the HSSP [75, 76].

Chapter 5

Portfolio Selection

The previous chapter was focused on the view of environmental selection in EAs as a subset selection problem. That work was aimed at addressing the limitations of one of the most popular methods for that purpose, the theoretically supported hypervolume indicator. This chapter takes a step back to rethink environmental selection, fitness assignment and DM preference integration as a portfolio selection problem (PSP), of which subset selection is a particular case. This new view started with the work by Yevseyeva *et al.* [133], and was materialized as a new indicator. Such work opened a path for a class of indicators modeled after portfolio selection problems, facilitating the expression of preferences, possibly without requiring as much computational effort as the hypervolume indicator. The mathematical formulation and the initial results let anticipate a new type of theoretically supported indicators.

The Hypervolume Sharpe-Ratio (HSR) indicator [133, 82] consists of maximizing the Sharpe ratio of a PSP resulting from a particular way of modeling uncertainty with respect to unknown DM preferences over the objective space. Other methods for modeling uncertainty would result in different indicators. Therefore, a class of Sharpe-ratio indicators can be formalized, and each uncertainty model results in a new instance of such a class. This chapter has two main goals. The first goal is to study the HSR indicator mainly theoretically but also experimentally, providing tools for the theoretical study of Sharpe ratio-based indicators. The second goal is to explore different instances of the Sharpe-ratio class by considering models expressed by different preference relations, in order to account for problems other then traditional (unconstrained) multiobjective optimization problems.

This chapter begins with the formalization of the class of Sharpe-ratio indicators (Section 5.2). The indicator proposed in the original paper [133] is then presented as an instance of this class in Section 5.3. Then, first-order necessary optimality (or Karush-Kuhn-Tucker) conditions are used to prove some theoretical properties of the HSR indicator in Section 5.4, while other characteristics of the indicator are only empirically studied in Section 5.5. The results are compared against the hypervolume indicator. The last two sections are dedicated to two new Sharpe-ratio indicator instances which extend the HSR indicator in two different ways. These extensions are aimed at covering two types of problems not contemplated by the HSR indicator namely set-valued optimization problems (in Section 5.6) and constrained multiobjective optimization problems (in Section 5.7).

5.1 Notation

Throughout the text, subscripts are used to refer to coordinates of points or vectors (e.g., a_i denotes the i^{th} coordinate of vector/point $a \in \mathbb{R}^d$), whereas bracketed superscripts will be used for enumeration (e.g., $a^{(1)}, a^{(2)}, a^{(3)} \in \mathbb{R}^d$). Moreover, in this chapter, the asterisk ("*") is used to denote optimality of a point (vector) or of a point set, which is different from the previous two chapters.

5.2 Sharpe-Ratio Indicator

The return of a solution depends on the DM preferences. As typically these preferences are not fully known, probabilistic models can be used to describe the associated uncertainty, such as the one described in Section 2.4.5. In general, different models of preference uncertainty lead to different random returns and, thus, to different allocations of investment when the Sharpe ratio is maximized (Problem 2.2). A broad class of indicators based on the Sharpe ratio can be defined as follows:

Definition 5.1 (Sharpe-Ratio Indicator). Given a non-empty set of assets $\mathrm{A} = \{a^{(1)}, \ldots, a^{(n)}\}$, the corresponding expected return, r, and covariance matrix, Q, the Sharpe-ratio indicator is given by

$$I_{\mathrm{SR}}(\mathrm{A}) = \max_{x \in \Omega} \; h^{\mathrm{A}}(x) \tag{5.1}$$

where $\Omega \subset [0,1]^n$ is the set of investment vectors that satisfy the constraints of Problem 2.2.

Note that the Sharpe-ratio indicator simultaneously evaluates the quality of a set in terms of a scalar, $I_{\mathrm{SR}}(\cdot)$, and, as a by-product, the relative importance of each solution in the set through the optimal investment vector x^*.

5.3 Hypervolume Sharpe-Ratio Indicator (HSR)

Let the Sharpe ratio $h^{\mathrm{A}}(x)$ for the set of solutions A where r and Q are defined as in (2.11) be represented by $h^{\mathrm{A}}_{\mathrm{HSR}}(x, l, u)$, and similarly for $g^{\mathrm{A}}(y)$ and $g^{\mathrm{A}}_{\mathrm{HSR}}(y, l, u)$. Analogously to the Sharpe-ratio indicator, the Hypervolume Sharpe-Ratio (HSR) indicator is formally defined as follows:

Definition 5.2 (Hypervolume Sharpe-Ratio Indicator). Given a non-empty point set $\mathrm{A} = \{a^{(1)}, \ldots, a^{(n)}\} \subset \mathbb{R}^d$, the points $l, u \in \mathbb{R}^d$ such that $l \ll u$, the expected return p and the covariance Q computed as in (2.11), the hypervolume Sharpe-ratio indicator $I_{\mathrm{HSR}}(\mathrm{A}, l, u)$ is given by:

$$I_{\mathrm{HSR}}(\mathrm{A}, l, u) = \max_{x \in \Omega} \; h^{\mathrm{A}}_{\mathrm{HSR}}(x, l, u) \tag{5.2}$$

where $\Omega \subset [0,1]^n$ is the set of investment vectors that satisfy the constraints of Problem 2.2.

5.3.1 Mathematical Analysis of the HSR Indicator

Computing the HSR indicator amounts to solving a convex quadratic programming problem (Problem 2.3). Furthermore, as Yevseyeva *et al.* [133] point out, it follows from the definition of q_{ij} that the return of the riskless asset under this model must be $r_f = 0$. Consequently, Problem 2.3 may be simplified by noting that constraint (2.10b) must always be satisfied. Therefore, the following is true for any feasible vector y:

$$y^T Q y = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} y_i y_j - \sum_{i=1}^{n} p_i y_i \sum_{j=1}^{n} p_j y_j = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} y_i y_j - 1 = y^T P y - 1 \quad (5.3)$$

Note that this simplification applies to any model of preference uncertainty as long as $r_f = 0$. Under these conditions, and adopting the notation and conventions of Nocedal and Wright [112], Problem 2.3 can be restated as follows:

Problem 5.1 (Sharpe-Ratio Maximization – Alternative QP formulation).

$$\min_{y \in \mathbb{R}^n} \quad y^T P y \quad (5.4a)$$

$$\text{subject to} \quad c_1(y) = \sum_{i=1}^{n} p_i y_i - 1 = 0 \quad (5.4b)$$

$$c_{i+1}(y) = y_i \geq 0, \ \ i = 1, \ldots, n \quad (5.4c)$$

This problem can be analyzed using the classical first-order necessary optimality conditions, or Karush-Kuhn-Tucker (KKT) conditions [112, p. 321]. Clearly, the objective function $y^T P y$ and all constraints are continuously differentiable. Since all constraints are linear, all active constraints must also be linear, which is a suitable constraint qualification [112, p. 338f]. The KKT conditions state that, if y^* is a locally optimal solution, and the above conditions hold, then there is a Lagrange multiplier vector λ^* such that all components associated with inactive constraints are zero (complementarity condition), and all components associated with inequality constraints are nonnegative. Moreover, the gradient with respect to y of the Lagrangian function at y^* is zero. This is denoted by $\nabla_y \mathcal{L}(y^*, \lambda^*) = 0$.

The Lagrangian function for Problem 5.1 is:

$$\mathcal{L}(y, \lambda) = y^T P y - \lambda_1 (p^T y - 1) - \sum_{i=1}^{n} \lambda_{i+1} y_i \quad (5.5)$$

and the corresponding partial derivatives with respect to y_k, $k = 1, \ldots, n$, are:

$$\frac{\partial \mathcal{L}(y, \lambda)}{\partial y_k} = 2 \sum_{i=1}^{n} p_{ki} y_i - p_k \lambda_1 - \lambda_{k+1} \quad (5.6)$$

Finally, note that Problems 2.3 and 5.1 have exactly the same optimal solution(s) by construction. Since Problem 2.3 is a convex quadratic programming problem, the KKT conditions are sufficient for a globally minimal solution, and the solution is unique if Q is positive definite [112, p. 464f]. By computing the gradient of the Lagrangian function of Problem 2.3 and noting that $p^T y = 1$ as in expression (5.3), it can be easily shown that any solution satisfying the KKT conditions of either problem also satisfies the KKT conditions of the other problem. Even though the value of Lagrange multiplier λ_1 is not the same in both cases, this is of no consequence as this multiplier is associated with an equality constraint. Therefore, the KKT conditions are also sufficient for a global solution of Problem 5.1.

5.4 Properties of the HSR Indicator

The rich mathematical structure of the HSR indicator allows its properties to be rigorously investigated. In the following, the optimal investment is shown to be invariant to the setting of the lower reference point l under certain (mild) conditions, but not to the setting of u. Varying l can also be interpreted as applying certain linear transformations to the objective space, under which the indicator is scaling independent. Weak $\preceq$-monotonicity and strict $\prec\cdot$-monotonicity of the HSR indicator are then proved. The derivation of the optimal μ-distributions on two-objective linear fronts concludes the present analysis.

5.4.1 Reference Points and Linear Scaling

The upper reference point, u, can have a significant impact in the HSR indicator. Note that the optimal (and only possible) allocation of investment to a single-point set is given by $x = (x_1) = 1$, and that the corresponding value of the HSR indicator is $\sqrt{p_1/(1-p_1)}$. Consequently, the higher p_1, the higher the indicator value.

As an example, consider two single-point sets $\mathrm{A} = \{(1,2)^T\}$ and $\mathrm{B} = \{(2,1)^T\}$, and a lower reference point, $l = (0,0)^T$. Setting the upper reference point $u = (3,4)^T$, $p_1^{\mathrm{A}} = 1/3$ and $p_1^{B} = 1/4$, which means that $I_{\mathrm{HSR}}(\mathrm{A}, l, u) > I_{\mathrm{HSR}}(\mathrm{B}, l, u)$. However, if the upper reference point is set to $u' = (4,3)^T$ instead, then $I_{\mathrm{HSR}}(\mathrm{A}, l, u') < I_{\mathrm{HSR}}(\mathrm{B}, l, u')$. Consequently, the upper reference point affects both the values of the HSR indicator and the order imposed by this indicator on sets. As it will be seen in Subsection 5.4.3, the choice of the upper reference point generally affects the optimal investment, as well.

In contrast, the location of the lower reference point, l, can be shown to have no effect on the optimal investment in a non-empty point set $\mathrm{A} \subset \mathbb{R}^d$ as long as u remains unchanged, and $a \in [l, u[\, = \{x \in \mathbb{R}^d \mid l \leq x \ll u\}$ holds true for all $a \in \mathrm{A}$. Moving l subject to these conditions is equivalent to applying a linear transformation to the objective space, with u as the center of the transformation. Thus, in practice, only one parameter of the HSR indicator needs to be set (the upper reference point, u). Formally:

Theorem 5.1. *Let $\mathrm{A} \subset \mathbb{R}^d$ be a non-empty point set, let $l, u \in \mathbb{R}^d$ be two reference points such that $\forall_{a \in \mathrm{A}},\ l \leq a \ll u$, and let $x^* \in [0,1]^n$ be such that $\sum_{i=1}^n x_i^* = 1$ and $I_{\mathrm{HSR}}(\mathrm{A}, l, u) = h_{\mathrm{HSR}}^{\mathrm{A}}(x^*, l, u)$. If $l' \in \mathbb{R}^d$ is such that $\{l'\} \preceq \mathrm{A}$, then x^* also satisfies $I_{\mathrm{HSR}}(\mathrm{A}, l', u) = h_{\mathrm{HSR}}^{\mathrm{A}}(x^*, l', u)$.*

Proof. Recall expression (2.11a) for the value of $p_{ij}(l, u)$ given a point set $\mathrm{A} = \{a^{(1)}, \ldots, a^{(n)}\} \subset \mathbb{R}^d$, and let $p(l, u) = [p_{ii}(l, u)]_{n \times 1}$ and $P(l, u) = [p_{ij}(l, u)]_{n \times n}$, where $i, j = 1, \dots, n$, as before. $P(l', u)$ can be related to $P(l, u)$ in the following way:

$$P(l', u) = \frac{v}{v'} P(l, u) \tag{5.7}$$

where $v = \Lambda([l, u])$ and $v' = \Lambda([l', u])$, and similarly for $p(l', u)$ and $p(l, u)$.

Consider an instance of Problem 2.3 where $r_f = 0$, $r = p(l', u)$ and $Q = P(l', u) - p(l', u)p(l', u)^T$, and let y' denote the vector of decision variables of that instance. Equality constraint (2.10b) can be rewritten as:

$$\begin{aligned} p(l', u)^T y' = 1 \quad &\Leftrightarrow \\ \frac{v}{v'} p(l, u)^T y' = 1 \quad &\Leftrightarrow \\ p(l, u)^T y = 1 \end{aligned} \tag{5.8}$$

by performing the change of variable $y = y'v/v'$. Similarly, with the same change of variable and the simplification pointed out in (5.3), the objective function (2.10a) can be rewritten as:

$$\begin{aligned} g^{\mathrm{A}}_{\mathrm{HSR}}(y', l', u) &= y'^T P(l', u) y' - 1 \\ &= y'^T \frac{v}{v'} P(l, u) y' - 1 \\ &= \frac{v'}{v} y^T P(l, u) y - 1 \\ &= \frac{v'}{v} g^{\mathrm{A}}_{\mathrm{HSR}}(y, l, u) + \frac{v'}{v} - 1 \end{aligned} \tag{5.9}$$

Since v'/v is positive, and the constraints on y and y' are equivalent, the constrained optimal solution y'^* of $g^{\mathrm{A}}_{\mathrm{HSR}}(y', l', u)$ can be obtained from the constrained optimal solution y^* of $g^{\mathrm{A}}_{\mathrm{HSR}}(y, l, u)$ simply by reverting the change of variable, i.e., $y'^* = y^* v'/v$. Consequently, the optimal investment vector

$$x^* = \frac{y^*}{\sum_{i=1}^n y^*_i} = \frac{y'^*}{\sum_{i=1}^n y'^*_i} \tag{5.10}$$

is the same in both cases, and the Theorem is proved. □

Note that changing some components of the lower reference point, l, while keeping the remaining ones fixed is equivalent to linearly scaling the corresponding objectives with respect to the remaining ones. Thus, the placement of l can also be seen as a way of linearly scaling the objective functions as long as this reference point continues to dominate all points in A. By Theorem 5.1, scaling the objective space under these conditions does not affect the optimal investment.

Scaling the objective space through l comes down to multiplying p_i and p_{ij} by a positive constant as in the above proof. Observing the Sharpe ratio expression $h(x)$ in Problem 2.2, the HSR indicator is not scaling invariant, i.e., scaling the objective space will affect the indicator value. However, the HSR indicator is scaling independent under these linear transformations, as shown next.

Theorem 5.2 (Linear-Scaling Independence of I_{HSR}). *Let $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$ be two non-empty point sets and $l, u \in \mathbb{R}^d$ be two reference points such that $\forall_{a \in \mathrm{A}, b \in \mathrm{B}}, l \leq a \ll u$ and $l \leq b \ll u$. If $I_{\mathrm{HSR}}(\mathrm{A}, l, u) \leq I_{\mathrm{HSR}}(\mathrm{B}, l, u)$, then $I_{\mathrm{HSR}}(\mathrm{A}, l', u) \leq I_{\mathrm{HSR}}(\mathrm{B}, l', u)$ holds true for any $l' \in \mathbb{R}^d$ such that $\{l'\} \preceq \mathrm{A} \cup \mathrm{B}$.*

Proof. Let p^{A}, P^{A} and Q^{A} denote, respectively, the expected return vector, the matrix of expected returns and the return covariance matrix associated with point set A given reference points l and u. Analogously, let p'^{A}, P'^{A} and Q'^{A} represent the same entities associated with point set A, but given reference points l' and u. As stated in (5.7), changing the lower reference point from l to l' results in the multiplication of the return probabilities by a constant factor $t = \Lambda([l,u])/\Lambda([l',u])$. In other words, $P'^{\mathrm{A}} = t\,P^{\mathrm{A}}$, which implies $p'^{\mathrm{A}} = t\,p^{\mathrm{A}}$, and similarly for set B.

Recall that the optimal investment, denoted below by x^{A} and x^{B}, is invariant to the choice of the lower reference point (Theorem 5.1), and note that both the numerator and the denominator of the Sharpe ratio of these portfolios must be strictly positive. Consequently,

$$\begin{aligned}
I_{\mathrm{HSR}}(\mathrm{A},l',u) &\le I_{\mathrm{HSR}}(\mathrm{B},l',u) && \Leftrightarrow\\
h^{\mathrm{A}}_{\mathrm{HSR}}(x^{\mathrm{A}},l',u) &\le h^{\mathrm{B}}_{\mathrm{HSR}}(x^{\mathrm{B}},l',u) && \Leftrightarrow\\
\frac{{p'^{\mathrm{A}}}^T x^{\mathrm{A}}}{\sqrt{{x^{\mathrm{A}}}^T Q'^{\mathrm{A}}\,x^{\mathrm{A}}}} &\le \frac{{p'^{\mathrm{B}}}^T x^{\mathrm{B}}}{\sqrt{{x^{\mathrm{B}}}^T Q'^{\mathrm{B}}\,x^{\mathrm{B}}}} && \Leftrightarrow\\
\frac{t\,{p^{\mathrm{A}}}^T x^{\mathrm{A}}}{\sqrt{t\,{x^{\mathrm{A}}}^T P^{\mathrm{A}}\,x^{\mathrm{A}} - t^2\,{x^{\mathrm{A}}}^T p^{\mathrm{A}}\,{p^{\mathrm{A}}}^T x^{\mathrm{A}}}} &\le \frac{t\,{p^{\mathrm{B}}}^T x^{\mathrm{B}}}{\sqrt{t\,{x^{\mathrm{B}}}^T P^{\mathrm{B}}\,x^{\mathrm{B}} - t^2\,{x^{\mathrm{B}}}^T p^{\mathrm{B}}\,{p^{\mathrm{B}}}^T x^{\mathrm{B}}}} && \Leftrightarrow \qquad (5.11)\\
\frac{{x^{\mathrm{A}}}^T p^{\mathrm{A}}\,{p^{\mathrm{A}}}^T x^{\mathrm{A}}}{(1/t)\,{x^{\mathrm{A}}}^T P^{\mathrm{A}}\,x^{\mathrm{A}} - {x^{\mathrm{A}}}^T p^{\mathrm{A}}\,{p^{\mathrm{A}}}^T x^{\mathrm{A}}} &\le \frac{{x^{\mathrm{B}}}^T p^{\mathrm{B}}\,{p^{\mathrm{B}}}^T x^{\mathrm{B}}}{(1/t)\,{x^{\mathrm{B}}}^T P^{\mathrm{B}}\,x^{\mathrm{B}} - {x^{\mathrm{B}}}^T p^{\mathrm{B}}\,{p^{\mathrm{B}}}^T x^{\mathrm{B}}} && \Leftrightarrow\\
\frac{1}{t}\left({x^{\mathrm{A}}}^T p^{\mathrm{A}}\,{p^{\mathrm{A}}}^T x^{\mathrm{A}}\right)\left({x^{\mathrm{B}}}^T P^{\mathrm{B}}\,x^{\mathrm{B}}\right) &\le \frac{1}{t}\left({x^{\mathrm{B}}}^T p^{\mathrm{B}}\,{p^{\mathrm{B}}}^T x^{\mathrm{B}}\right)\left({x^{\mathrm{A}}}^T P^{\mathrm{A}}\,x^{\mathrm{A}}\right)
\end{aligned}$$

Since the constant t vanishes from the inequality, which includes the case where the lower reference point is not changed ($t = 1$), the Theorem is proved. □

5.4.2 Monotonicity

Monotonicity properties can be stated for the HSR indicator by building upon the following auxiliary results. As in expression (2.11), let p_{ij} denote the probability that two points $a^{(i)}, a^{(j)} \in \mathbb{R}^d$ are both acceptable to the same DM for some integer i and j, and let $p_i = p_{ii}$.

Lemma 5.3. *Suppose that two points $a^{(1)}, a^{(2)} \in \mathbb{R}^d$ and two reference points $l, u \in \mathbb{R}^d$ are such that $l \le a^{(2)} < a^{(1)} \ll u$. Then, for all $a^{(3)} \in [l,u[$, the following holds true:*

$$p_1 p_{23} \le p_{13} p_2 \qquad (5.12)$$

Proof. Without loss of generality, consider that $l = (0,\dots,0)$ and $u = (1,\dots,1)$ and, therefore, $\Lambda([l,u]) = 1$. Assume that, for some choice of $a^{(3)} \in [l,u[$,

$$\begin{aligned}
p_1 p_{23} \quad &> \quad p_{13} p_2 && \Leftrightarrow\\
\textstyle\prod_{k=1}^{d} \left(1 - a_k^{(1)}\right)\left(1 - \max\left(a_k^{(2)}, a_k^{(3)}\right)\right) \quad &> \quad \textstyle\prod_{i=1}^{d} \left(1 - \max\left(a_k^{(1)}, a_k^{(3)}\right)\right)\left(1 - a_k^{(2)}\right)
\end{aligned} \qquad (5.13)$$

Clearly, if $a_k^{(i)} = 1$ for some $i \in \{1, 2, 3\}$ and k, the inequality is false. Otherwise, there should be at least a dimension $k \in \{1, \dots, d\}$ for which the following holds true:

$$\left(1 - a_k^{(1)}\right)\left(1 - \max\left(a_k^{(2)}, a_k^{(3)}\right)\right) \;>\; \left(1 - \max\left(a_k^{(1)}, a_k^{(3)}\right)\right)\left(1 - a_k^{(2)}\right) \tag{5.14}$$

If $a_k^{(1)} \geq a_k^{(3)}$, it follows that:

$$\begin{aligned} \left(1 - a_k^{(1)}\right)\left(1 - \max\left(a_k^{(2)}, a_k^{(3)}\right)\right) &> \left(1 - a_k^{(1)}\right)\left(1 - a_k^{(2)}\right) \quad \Leftrightarrow \\ a_k^{(2)} &> \max\left(a_k^{(2)}, a_k^{(3)}\right) \end{aligned} \tag{5.15}$$

which is clearly false. Otherwise, $a_k^{(3)} > a_k^{(1)} \geq a_k^{(2)}$, and expression (5.14) is equivalent to

$$\begin{aligned} \left(1 - a_k^{(1)}\right)\left(1 - a_k^{(3)}\right) &> \left(1 - a_k^{(3)}\right)\left(1 - a_k^{(2)}\right) \quad \Leftrightarrow \\ a_k^{(2)} &> a_k^{(1)} \end{aligned} \tag{5.16}$$

which contradicts the assumption that $a^{(2)} < a^{(1)}$. Therefore, expression (5.13) is false, and the Lemma is proved. □

Lemma 5.4. *Let $l, u \in \mathbb{R}^d$ be two reference points such that $l \ll u$, and let $\mathrm{A} = \{a^{(1)}, \dots, a^{(n)}\} \subset [l, u[$, where $n \geq 2$, be a point set. If $a^{(2)} < a^{(1)}$, then the investment vector $x^* \in [0, 1]^n$ that maximizes the hypervolume Sharpe ratio for set A is such that $x_1^* = 0$.*

Proof. Consider the alternative QP formulation of Sharpe-ratio maximization given in Problem 5.1, and recall that if y^* denotes an optimal solution of this problem, $x^* = y^* / \sum_{i=1}^n y_i^*$ is an optimal solution of Problem 2.2.

By contradiction, assume that $y_1^* > 0$. By the KKT conditions, $\lambda_2^* = 0$ (complementarity) and

$$\frac{\partial \mathcal{L}(y^*, \lambda^*)}{\partial y_k} = 2 \sum_{i=1}^n p_{ki} y_i^* - p_k \lambda_1^* - \lambda_{k+1}^* = 0 \tag{5.17}$$

for all $k = 1, \dots, n$. Considering only the first two partial derivatives ($k = 1, 2$), this implies:

$$p_1 \frac{\partial \mathcal{L}(y^*, \lambda^*)}{\partial y_2} - p_2 \frac{\partial \mathcal{L}(y^*, \lambda^*)}{\partial y_1} = 0 \quad \Leftrightarrow \tag{5.18}$$

$$2 \sum_{i=1}^n (p_1 p_{2i} - p_{1i} p_2)\, y_i^* = p_1 \lambda_3^* \tag{5.19}$$

To be a valid Lagrange multiplier, $\lambda_3^* \geq 0$. However, by Lemma 5.3, no term of the sum can be positive, so neither can the sum itself. Moreover, the first term of the sum is clearly negative, since $p_{21} = p_1 < p_2$ due to $a^{(2)} < a^{(1)}$. Therefore, no Lagrange multiplier vector λ^* exists for which the KKT conditions are satisfied when $y_1^* > 0$, and consequently such a y^* cannot be optimal. Since $y_1^* = 0$ implies $x_1^* = 0$ as noted above, the Lemma is proved. □

For suitable choices of reference points, the HSR indicator can now be shown to be both weakly $\preceq$-monotonic and strictly $\prec\cdot$-monotonic, but not strictly $\prec$-monotonic.

Theorem 5.5 (Weak $\preceq$-Monotonicity of the HSR Indicator). *Given two reference points $l, u \in \mathbb{R}^d$ such that $l \ll u$ and two non-empty point sets $\mathrm{A}, \mathrm{B} \subset [l, u[$ such that $\mathrm{A} \preceq \mathrm{B}$, $I_{\mathrm{HSR}}(\mathrm{A}, l, u) \geq I_{\mathrm{HSR}}(\mathrm{B}, l, u)$.*

Proof. Consider $I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B})$. Since $\mathrm{A} \preceq \mathrm{B}$, any points in $\mathrm{B} \setminus \mathrm{A}$ are dominated points in $\mathrm{A} \cup \mathrm{B}$, and, by Lemma 5.4, are assigned zero investment. Therefore, $I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B}) = I_{\mathrm{HSR}}(\mathrm{A})$ must hold true. Assume that $I_{\mathrm{HSR}}(\mathrm{B}) > I_{\mathrm{HSR}}(\mathrm{A})$. Then, an investment strategy in $\mathrm{A} \cup \mathrm{B}$ with a Sharpe ratio greater than $I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B})$ exists (whereby all points in $\mathrm{A} \setminus \mathrm{B}$ are given zero investment), which is a contradiction and proves the Theorem. □

Theorem 5.6 (Strict $\prec\cdot$-Monotonicity of the HSR Indicator). *Given two reference points $l, u \in \mathbb{R}^d$ such that $l \ll u$ and two non-empty point sets $\mathrm{A}, \mathrm{B} \subset [l, u[$ such that $\mathrm{A} \prec\cdot \mathrm{B}$, $I_{\mathrm{HSR}}(\mathrm{A}, l, u) > I_{\mathrm{HSR}}(\mathrm{B}, l, u)$.*

Proof. As before, consider $I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B})$. Since $\mathrm{A} \prec\cdot \mathrm{B}$, all points in B are dominated points in $\mathrm{A} \cup \mathrm{B}$, and, therefore, $I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B}) = I_{\mathrm{HSR}}(\mathrm{A})$. Assume that $I_{\mathrm{HSR}}(\mathrm{B}) \geq I_{\mathrm{HSR}}(\mathrm{A})$. Then, an investment strategy in $\mathrm{A} \cup \mathrm{B}$ with a Sharpe ratio greater than or equal to $I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B})$ exists whereby all points in $\mathrm{A} \setminus \mathrm{B}$ are given zero investment, and strictly positive investment is given to at least one point in B. Since such a point is strictly dominated in $\mathrm{A} \cup \mathrm{B}$, a contradiction arises due to Lemma 5.4, and the Theorem is proved. □

Note that the empty set is excluded in the above theorems because the HSR indicator is not defined for the empty set (see Definition 5.2). However, if $I_{\mathrm{HSR}}(\emptyset, l, u) \triangleq 0$, for example, the same properties can be easily shown to apply also when the empty set is considered.

Lemma 5.7 (Lack of Strict $\prec$-Monotonicity of the HSR Indicator). *Given two reference points $l, u \in \mathbb{R}^d$ such that $l \ll u$, there are at least two point sets $\mathrm{A}, \mathrm{B} \subset [l, u[$ such that $\mathrm{A} \prec \mathrm{B}$ and $I_{\mathrm{HSR}}(\mathrm{A}, l, u) \leq I_{\mathrm{HSR}}(\mathrm{B}, l, u)$.*

Proof. Let $l = (0, 0)$, $u = (1, 1)$, and consider the sets $\mathrm{A} = \{a^{(1)}, a^{(2)}, a^{(3)}\}$ and $\mathrm{B} = \{a^{(1)}, a^{(3)}\}$, where $a^{(1)} = (0, 1-a)$, $a^{(2)} = (1-b, 1-b)$, $a^{(3)} = (1-a, 0)$, and $0 < a < b \leq 2a/(a+1) < 1$. Then, $y = [1/(2a), 0, 1/(2a)]^T$ satisfies the KKT conditions of Problem 5.1 for set A with corresponding Lagrange multiplier vector $\lambda = [\lambda_1, 0, \lambda_3, 0]^T$, where $\lambda_1 = 1 + 1/a$ and $\lambda_3 = 2b - \lambda_1 b^2 \geq 0$. Since the optimal investment in point $(1-b, 1-b)$ is zero, it follows that $I_{\mathrm{HSR}}(\mathrm{A}, l, u) = I_{\mathrm{HSR}}(\mathrm{B}, l, u)$ despite the fact that $\mathrm{A} \prec \mathrm{B}$. □

The strict $\prec\cdot$-monotonicity property of the HSR indicator guarantees, by Lemma 2.2, that at least one point in an indicator-optimal subset must lie on the Pareto front of the original set. The following result leads to a slightly stronger statement.

Lemma 5.8. *Let $l, u \in \mathbb{R}^d$ be two reference points such that $l \ll u$, let $\mathrm{A} = \{a^{(1)}, \ldots, a^{(n)}\} \subset [l, u[$, and let $\mathrm{P} = \mathsf{nondominated}(\mathrm{A})$. If $|\mathrm{P}| \geq 2$, the optimal solution, y^*, of Problem 5.1 given A, l and u is such that at least two points in P are assigned strictly positive investment.*

Proof. The equality constraint $c_1(y) = \sum_{i=1}^{n} p_i y_i = 1$ implies that $y_i^* > 0$ for at least one $i \in \{1, \ldots, n\}$. Also, by Lemma 5.4, all points in $\mathrm{A} \setminus \mathrm{P}$ are assigned zero investment.

By contradiction, assume without loss of generality that $a^{(1)}, a^{(2)} \in \mathrm{P}$, $y_1^* > 0$, and $y_j^* = 0$ for all $j = 2, \ldots, n$. Since constraint $c_2(y) = y_1 \geq 0$ is not active, the associated Lagrange multiplier, λ_2^*, must be zero for the KKT conditions to hold. Furthermore, λ_3^* must be nonnegative due to constraint $c_3(y) = y_2 \geq 0$ being active. Taking the partial derivatives of the Lagrangian function (5.5) with respect to y_1 and y_2, it follows that:

$$\begin{cases} 2\sum_{i=1}^{n} p_{i1} y_i^* - p_1 \lambda_1^* &= 0 \\ 2\sum_{i=1}^{n} p_{i2} y_i^* - p_2 \lambda_1^* - \lambda_3^* &= 0 \end{cases} \Leftrightarrow \begin{cases} 2p_1 y_1^* - p_1 \lambda_1^* &= 0 \\ 2p_{12} y_1^* - p_2 \lambda_1^* - \lambda_3^* &= 0 \end{cases}$$
$$\Leftrightarrow \begin{cases} \lambda_1^* &= 2y_1^* \\ \lambda_3^* &= 2p_{12} y_1^* - 2p_2 y_1^* \end{cases} \Leftrightarrow \begin{cases} — \\ \lambda_3^* &= 2y_1^*(p_{12} - p_2) < 0 \end{cases} \tag{5.20}$$

Note that $p_{12} < p_2$ because $a^{(1)}$ and $a^{(2)}$ are incomparable to each other. Since $y_1^* > 0$ and $(p_{12} - p_2) < 0$, λ_3^* is negative, and y^* cannot be optimal. $\square$

Theorem 5.9. *Let $l, u \in \mathbb{R}^d$ be two reference points such that $l \ll u$, let $\mathrm{S} \subset [l, u[$ be a non-empty point set, let $\mathrm{P} = \textsf{nondominated}(\mathrm{S})$, and let $k \geq 2$ be a positive integer. If $|\mathrm{P}| \geq 2$ and $\mathrm{A}_k^* \subset \mathrm{S}$ is an optimal subset of S with respect to I_{HSR} given k, then $|\mathrm{A}_k^* \cap \mathrm{P}| \geq 2$.*

Proof. Consider a non-empty indicator-optimal subset $\mathrm{B} \subset \mathrm{S}$ such that $|\mathrm{B}| \leq k$. Since $\mathrm{B} \cap \mathrm{P}$ cannot be empty by Lemma 2.2, assume that $\mathrm{B} \cap \mathrm{P} = \{b\}$. Consequently, for each point in $\mathrm{B} \setminus \{b\}$, there is a point in P that strictly dominates it. Let $\mathrm{A} \subseteq \mathrm{P}$ be a subset of points constructed by adding to $\{b\}$, for each point in $\mathrm{B} \setminus \{b\}$, one point in P that strictly dominates it, and then adding one more point from P not yet in A, if needed, to ensure that $2 \leq |\mathrm{A}| \leq k$. Since $\mathrm{A} \preceq \mathrm{B}$ by construction, $I_{\mathrm{HSR}}(\mathrm{A}) \geq I_{\mathrm{HSR}}(\mathrm{B})$ by Theorem 5.5. Furthermore, since any points in $\mathrm{B} \setminus \mathrm{A}$ are dominated points in $\mathrm{A} \cup \mathrm{B}$, by Lemma 5.4 they are assigned zero investment, and consequently $I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B}) = I_{\mathrm{HSR}}(\mathrm{A})$. Finally, by Lemma 5.8, at least two points in $\mathrm{A} \cup \mathrm{B}$, of which at most one may also be in B, will be assigned positive investment when computing $I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B})$. This implies that $I_{\mathrm{HSR}}(\mathrm{A}) = I_{\mathrm{HSR}}(\mathrm{A} \cup \mathrm{B}) > I_{\mathrm{HSR}}(\mathrm{B})$, which contradicts the assumption that B is indicator-optimal. $\square$

5.4.3 Optimal μ-Distributions on Two-Objective Linear Fronts

Although the HSR indicator is not strictly $\prec$-monotonic, its weak $\preceq$-monotonicity implies, by Lemma 2.1, that there exists at least one indicator-optimal set of up to a given size that is a subset of the Pareto front. Moreover, by Lemma 5.4, all HSR-optimal subsets are such that all points that contribute to the indicator value are nondominated with respect to the original set. Therefore, only points on the

Pareto front need to be considered in the study of the optimal μ-distributions of the HSR indicator. Here, that study is further restricted to bounded continuous fronts in the two-objective case and, in particular, to linear fronts.

Definition 5.3 (Two-Objective Bounded Continuous Nondominated Front). Consider $s^{(1)}, s^{(2)} \in \mathbb{R}$ such that $s^{(1)} < s^{(2)}$, and let $f : [s^{(1)}, s^{(2)}] \to \mathbb{R}$ be a bounded, continuous, strictly decreasing function. Then, $\mathrm{X} = \{(v, f(v)) \mid s^{(1)} \leq v \leq s^{(2)})\}$ is a two-objective bounded continuous nondominated front. If f is a linear function, X is also called a two-objective (bounded, continuous) linear front.

Problem 5.1 can now be reformulated to include the position of the points on a given front as variables.

Problem 5.2 (μ-Distribution Optimization). Given two reference points $l, u \in \mathbb{R}^2$ such that $l \ll u$, an integer $\mu = n > 0$, and a (bounded) continuous nondominated front $\mathrm{X} \subset [l, u]$, defined according to Definition 5.3, such that at least one point in X strongly dominates u, find a globally optimal solution, (y^*, v^*), of:

$$\min_{y\in\mathbb{R}^n, v\in\mathbb{R}^n} \quad g(y, v) = y^T P y \tag{5.21a}$$

$$\text{subject to} \quad c_1(y, v) = \sum_{i=1}^{n} p_i y_i - 1 = 0 \tag{5.21b}$$

$$c_2(y, v) = -s^{(1)} + v_1 \geq 0 \tag{5.21c}$$

$$c_3(y, v) = s^{(2)} - v_n \geq 0 \tag{5.21d}$$

$$c_j(y, v) = y_i \geq 0, \qquad j = i + 3,\ i = 1, \dots, n \tag{5.21e}$$

$$c_j(y, v) = -v_i + v_{i+1} \geq 0, \qquad j = i + 3 + n,\ i = 1, \dots, n - 1 \tag{5.21f}$$

where $v = (v_1, \dots, v_n)$ represents the x-coordinates of n points in X, $P = [p_{ij}]_{n\times n}$, the expected return $p_i = p_{ii}$, and, for $i \leq j$,

$$p_{ij} = p_{ji} = \frac{(u_2 - f(v_i))(u_1 - v_j)}{(u_2 - l_2)(u_1 - l_1)} \tag{5.22}$$

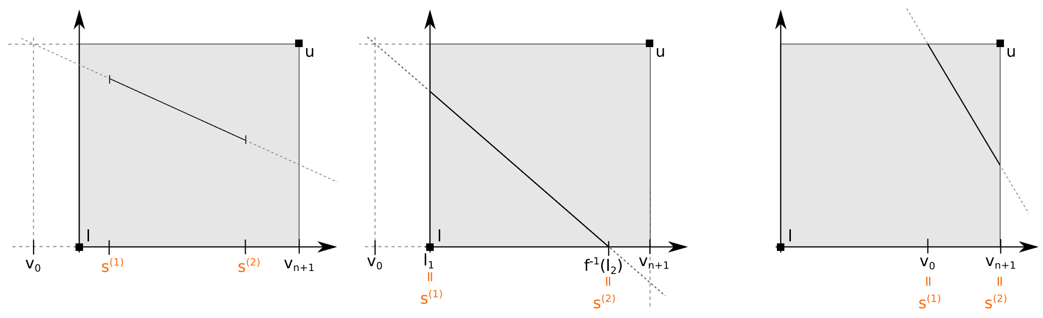
The set $\{(v_i^*, f(v_i^*)) \mid i = 1, \dots, n\}$ is an optimal μ-distribution of the HSR indicator on X, and $x^* = y^* / \sum_{i=1}^n y_i^*$ is the corresponding optimal investment.

Constraints (5.21c) and (5.21d) guarantee that all points $(v_i, f(v_i))$ in a feasible solution of the problem are contained in the nondominated front X, and constraints (5.21f) ensure that the n points are sorted in ascending order of their first coordinate.

Like Problem 5.1, this problem can be analyzed using the KKT conditions. Provided that f is continuously differentiable, so are the objective function and all constraints. However, the objective function is no longer a convex quadratic function, and constraint c_1 is no longer linear, due to the dependence of p_i (and p_{ij}) on v_i (and v_j), which considerably complicates the analysis. For this reason, only *linear* fronts are considered in the remaining of this work. Illustrative examples of linear fronts are given in Figure 5.1.

Nondegeneracy

The analysis of Problem 5.2 begins with the observation that any candidate solution at which any of the constraints (5.21e) and (5.21f) are active, or such that any of the



(a) $s^{(1)} > l_1, s^{(2)} < u_1$ (b) $s^{(1)} = l_1$, $s^{(2)} = f^{-1}(l_2)$ (c) $s^{(1)} = f^{-1}(u_2)$, $s^{(2)} = u_1$

Figure 5.1: Examples of linear fronts and the corresponding $s^{(1)}$ and $s^{(2)}$.

expected returns $p_i = 0$, $i = 1, \ldots, n$, is *degenerate* in the sense that it is equivalent to another solution with fewer than μ distinct points, all of which receiving positive investment.

Lemma 5.10. *Let* X *be a linear front such that $f(v) = -mv + b$, where $m, b \in \mathbb{R}$ and $m > 0$, and let $v_0 = (b - u_2)/m$ and $v_{n+1} = u_1$. Then, any globally optimal solution (y^*, v^*) of Problem 5.2 with $\mu = n$ points is such that $v_0 < v_i^* < v_{i+1}^* < v_{n+1}$ for all $i = 1, \ldots, n-1$, and $y_i^* > 0$ for all $i = 1, \ldots, n$.*

Proof. The proof is made by induction on the number of points, μ.

Base case 1 When $\mu = 1$, any solution such that $v_1 = v_0$ or $v_1 = v_2$ is infeasible, as it corresponds to $p_1 = 0$, and violates constraint c_1. Therefore, any globally optimal (y^*, v^*) must be such that $v_0 < v_1^* < v_2$ and $y_1^* > 0$.

Base case 2 For $\mu = 2$, let (y, v) denote a feasible candidate solution. The following assumptions are equivalent in the sense that they lead to degenerate solutions that correspond to a single point set, $\{(v_1, f(v_1))\}$, where v_1 must obey $v_0 < v_1 < v_3$:

- $y_2 = 0$
- $v_2 = v_i$, $i = 0, 1, 3$

By Lemma 5.8, under any of the above assumptions on (y, v) and for any $v_2' \neq v_1$ such that $v_0 < v_2' < v_3$, there is a feasible solution (y', v') such that $v_1' = v_1$, $y_1' > 0$, $y_2' > 0$, and $y'^T P y' < y^T P y$. Therefore, any globally optimal solution (y^*, v^*) for $\mu = 2$ must also be nondegenerate.

Inductive step To show that any globally optimal solution of Problem 5.2 for $\mu = n \geq 3$ must be nondegenerate under the hypothesis that this is true for $\mu = n - 1$, let (y, v) denote a feasible solution for the case where $\mu = n$ such that, for some integer $j \in \{2, \ldots, n-1\}$:

$$\begin{aligned} &y_i > 0 \quad \text{for all} \quad i = 1, \ldots, j-1, j+1, \ldots, n \\ &v_0 < \cdots < v_{j-1} < v_{j+1} < \cdots < v_{n+1} \end{aligned} \tag{5.23}$$

The following assumptions on (y, v) are equivalent in the sense that they lead to degenerate solutions that correspond to the same $(n-1)$-point set:

- $y_j = 0$
- $v_j = v_i$, $i = 0, \dots, j-1, j+1, \dots, n+1$

Under any of these assumptions, $y^{(n-1)} = (\bar{y}_1, \dots, \bar{y}_{j-1}, \bar{y}_{j+1}, \dots, \bar{y}_n)$, where $\bar{y}_i = y_i$ if $v_i \neq v_j$ and $\bar{y}_i = y_i + y_j$ if $v_i = v_j$, and $v^{(n-1)} = (v_1, \dots, v_{j-1}, v_{j+1}, \dots, v_n)$ represent a nondegenerate feasible solution $(y^{(n-1)}, v^{(n-1)})$ for the case where $\mu = n-1$.

Without loss of generality, consider the case where $y_j = 0$ and, for convenience, let $v_j = (v_{j-1} + v_{j+1})/2$. Under the induction hypothesis, if $(y^{(n-1)}, v^{(n-1)})$ were itself degenerate, it could not be optimal for the case where $\mu = n-1$. Consequently, (y, v) could not be optimal for the case where $\mu = n$ either, because $y^T P y = {y^{(n-1)}}^T P^{(n-1)} y^{(n-1)}$, where $P^{(n-1)}$ denotes the matrix obtained from P by removing row and column j.

Proceed by assuming that y is an optimal solution of Problem 5.1 given the set of points $\{(v_i, f(v_i)) \mid i = 1, \dots, n\}$. Then, there must be a Lagrange multiplier vector λ such that $\lambda_{j+1} \geq 0$, reflecting the fact that constraint $c_{j+1}(y) = y_j \geq 0$ is active, and $\lambda_j = \lambda_{j+2} = 0$, as the corresponding constraints are not active.

The KKT condition on the gradient of the Lagrangian function implies that:

$$\begin{aligned}
\frac{\partial \mathcal{L}(y,\lambda)}{\partial y_{j-1}} + \frac{\partial \mathcal{L}(y,\lambda)}{\partial y_{j+1}} &= 2\sum_{i=1}^{n} \left(p_{(j-1)i} + p_{(j+1)i}\right) y_i - \left(p_{j-1} + p_{j+1}\right)\lambda_1 &&= 0 \\
&\Leftrightarrow \frac{2\sum_{i=1}^{n} \left(p_{(j-1)i} + p_{(j+1)i}\right) y_i}{p_{j-1} + p_{j+1}} &&= \lambda_1
\end{aligned} \tag{5.24}$$

Plugging λ_1 above into the condition on the partial derivative of the Lagrangian function with respect to y_j, it follows that:

$$\begin{aligned}
\frac{\partial \mathcal{L}(y,\lambda)}{\partial y_j} &= 2\sum_{i=1}^{n} p_{ji} y_i - p_j \lambda_1 - \lambda_{j+1} &&= 0 \\
&\Leftrightarrow 2\sum_{i=1}^{n} p_{ji} y_i - \frac{2\sum_{i=1}^{n} \left(p_{(j-1)i} + p_{(j+1)i}\right) y_i}{p_{j-1} + p_{j+1}} p_j - \lambda_{j+1} &&= 0 \\
&\Leftrightarrow \frac{2}{p_{j-1} + p_{j+1}} \sum_{i=1}^{n} \left[(p_{j-1} + p_{j+1}) p_{ji} - \left(p_{(j-1)i} + p_{(j+1)i}\right) p_j\right] y_i &&= \lambda_{j+1}
\end{aligned} \tag{5.25}$$

Since $y_j = 0$, the term

$$\left[(p_{j-1} + p_{j+1}) p_{ji} - \left(p_{(j-1)i} + p_{(j+1)i}\right) p_j\right] y_i \tag{5.26}$$

is 0 for $i = j$. Recalling that $v_j = (v_{j-1} + v_{j+1})/2$, $p_i = p_{ii}$, and that, for $i \leq j$:

$$\begin{aligned}
p_{ij} = p_{ji} &= \frac{(u_2 - f(v_i))(u_1 - v_j)}{(u_2 - l_2)(u_1 - l_1)} \\
&= \frac{m\,(v_i - v_0)(v_{n+1} - v_j)}{(u_2 - l_2)(u_1 - l_1)}
\end{aligned} \tag{5.27}$$

the remaining terms are equal to:

$$-\frac{m^2(v_{j+1}-v_{j-1})^2(2v_{n+1}-v_{j-1}-v_{j+1})(v_i-v_0)\,y_i}{4\,(u_2-l_2)^2(u_1-l_1)^2} \qquad \text{if } i<j \tag{5.28}$$

$$-\frac{m^2(v_{j+1}-v_{j-1})^2(v_{j-1}+v_{j+1}-2v_0)(v_{n+1}-v_i)\,y_i}{4\,(u_2-l_2)^2(u_1-l_1)^2} \qquad \text{if } i>j \tag{5.29}$$

Under the assumptions (5.23) placed on (y, v), all of these terms are strictly negative, and therefore so is λ_{j+1}. Consequently, y is not an optimal solution of Problem 5.1, and there is a nondegenerate feasible solution (y', v) of Problem 5.2 at which $y'^T Py' < y^T Py$. Therefore, any globally optimal solution (y^*, v^*) for $\mu = n$ must also be nondegenerate.

□

By Lemma 5.10, none of the constraints c_4 to c_{2n+2} of Problem 5.2 are active at the optimum when X is a linear front, and the corresponding Lagrange multipliers λ_4 to λ_{2n+2} must be equal to zero. In contrast, inequality constraints c_2 and c_3 may be active, and equality constraint c_1 is necessarily active, but the gradients of these constraints with respect to (y, v) are clearly linearly independent for all nondegenerate solutions, where all components of $\nabla_y c_1(y, v) = p$ are positive. Consequently, the Linear Independence Constraint Qualification [112, p. 320] holds at those solutions, and the Lagrangian function for Problem 5.2 can be written as:

$$\mathcal{L}(y, v, \lambda) = \; y^T Py - \lambda_1 \left(\textstyle\sum_{i=1}^n p_i y_i - 1\right) - \lambda_2 \left(-s^{(1)} + v_1\right) - \lambda_3 \left(s^{(2)} - v_n\right) \tag{5.30}$$

Optimal investment

The partial derivatives of the Lagrangian function with respect to y_k, $k = 1, \ldots, n$, are:

$$\frac{\partial \mathcal{L}(y, v, \lambda)}{\partial y_k} = 2\sum_{i=1}^{n} p_{ki} y_i - p_k \lambda_1 \tag{5.31}$$

For a given v, setting these partial derivatives to zero and satisfying constraint c_1 leads to a system of $n+1$ linear equations and $n+1$ variables, which can be written in matrix form as follows:

$$\left[\begin{array}{c|c} 2P & -p \\ \hline p^T & 0 \end{array}\right] \left[\begin{array}{c} y \\ \hline \lambda_1 \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 1 \end{array}\right] \tag{5.32}$$

Considering again a linear front such that $f(v) = -mv + b$, $m > 0$, and the corresponding values $p_i = p_{ii}$ and $p_{ij} = p_{ji}$ obtained in expression (5.27), solving the above system leads to:

$$y_i = \frac{v_{i+1} - v_{i-1}}{w} \tag{5.33a}$$

$$\lambda_1 = \frac{2(v_{n+1} - v_0)}{w} \tag{5.33b}$$

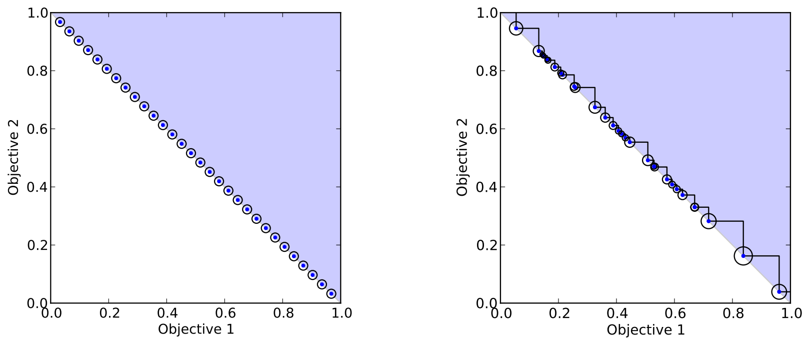


Figure 5.2: Optimal investment for two sets of 30 points on a linear front. Left: equally spaced points. Right: randomly placed points. Circle area is proportional to assigned investment. $l = (0, 0)$ and $u = (1, 1)$.

for $i = 1, \ldots, n$, where $v_0 = (b - u_2)/m$, $v_{n+1} = u_1$, and

$$w = \frac{m}{(u_2 - l_2)(u_1 - l_1)} \sum_{i=1}^{n} (v_{n+1} - v_i)\,(v_{i+1} - v_{i-1})\,(v_i - v_0) \tag{5.34}$$

is clearly positive for any nondegenerate solution. Consequently, λ_1 is positive, too. In addition, since

$$\sum_{i=1}^{n} y_i = \frac{1}{w} \sum_{i=1}^{n} (v_{i+1} - v_{i-1}) = \frac{v_{n+1} - v_0 + v_n - v_1}{w} \tag{5.35}$$

the corresponding (optimal) investment vector x is such that:

$$x_i = \frac{v_{i+1} - v_{i-1}}{v_{n+1} - v_0 + v_n - v_1} \tag{5.36}$$

for all $i = 1, \ldots, n$. This means that the optimal investment in each point on a linear front must be proportional to the distance between its two direct neighbours regardless of its position between those neighbours. This is reminiscent of the crowding distance in NSGA II [53], and confirms what was experimentally observed by Yevseyeva *et al.* [133] (see Figure 5.2).

Optimal placement

The partial derivatives of the Lagrangian function with respect to the components of v are:

$$\frac{\partial \mathcal{L}(y, v, \lambda)}{\partial v_1} = y_1^2 \frac{\partial p_1}{\partial v_1} + 2 \sum_{k=2}^{n} y_1 y_k \frac{\partial p_{1k}}{\partial v_1} - \lambda_1 y_1 \frac{\partial p_1}{\partial v_1} - \lambda_2 \tag{5.37a}$$

$$\frac{\partial \mathcal{L}(y, v, \lambda)}{\partial v_i} = y_i^2 \frac{\partial p_i}{\partial v_i} + 2 \sum_{\substack{k=1 \\ k \neq i}}^{n} y_i y_k \frac{\partial p_{ik}}{\partial v_i} - \lambda_1 y_i \frac{\partial p_i}{\partial v_i} \tag{5.37b}$$

$$\frac{\partial \mathcal{L}(y, v, \lambda)}{\partial v_n} = y_n^2 \frac{\partial p_n}{\partial v_n} + 2 \sum_{k=1}^{n-1} y_n y_k \frac{\partial p_{nk}}{\partial v_n} - \lambda_1 y_n \frac{\partial p_n}{\partial v_n} + \lambda_3 \tag{5.37c}$$

for $i = 2, \dots, n-1$. Plugging expressions (5.33a) and (5.33b) into the above partial derivatives eliminates y and λ_1 from them. Setting the resulting expressions to zero leads to a system of n nonlinear equations and $n+2$ variables that simplifies to the following equations:

$$v_i - v_{i-1} = v_{i+1} - v_i, \qquad i = 2, \dots, n-1 \tag{5.38a}$$

$$(v_1 - v_0) - (v_2 - v_1) = \lambda_2 \, \frac{w^2(u_2 - l_2)(u_1 - l_1)}{m\,(v_2 - v_0)(v_{n+1} - v_0)} \tag{5.38b}$$

$$(v_{n+1} - v_n) - (v_n - v_{n-1}) = \lambda_3 \, \frac{w^2(u_2 - l_2)(u_1 - l_1)}{m\,(v_{n+1} - v_{n-1})(v_{n+1} - v_0)} \tag{5.38c}$$

The above conditions imply that an optimal solution (y^*, v^*) of Problem 5.2 must be such that the inner points v_2^* to v_{n-1}^* are evenly spaced between the two outer points, v_1^* and v_n^*. As a consequence, by expressions (5.33a) and (5.36), all inner points receive equal investment. In turn, the outer points v_1^* and v_n^* depend on the front end points, $s^{(1)}$ and $s^{(2)}$, and on v_0 and v_{n+1}.

Since the Lagrange multipliers λ_2^* and λ_3^* associated with an optimal solution must not be negative, and the fractions in the right-hand sides of equations (5.38b) and (5.38c) are clearly positive, the optimal placement of the outer points v_1^* and v_n^* is such that the distances from v_1^* to v_0 and from v_n^* to v_{n+1} are greater than or equal to the distances from v_1^* to v_2^* and from v_n^* to v_{n-1}^*, respectively. Consequently, $v_1^* \geq (v_0 + v_2^*)/2$ and $v_n^* \leq (v_{n-1}^* + v_{n+1})/2$, and the optimal investment in each of the outer points, v_1^* and v_n^*, is greater than or equal to the investment in any of the inner points.

The actual value of v^* can now be derived by considering equations (5.38) together with the problem constraints c_2 and c_3. Supposing that both c_2 and c_3 are not active, the Lagrange multipliers λ_2^* and λ_3^* are zero, and it follows from equations (5.38) that:

$$v_i^* = v_0 + i\,\frac{v_{n+1} - v_0}{n+1}, \qquad i = 1, \dots, n \tag{5.39}$$

Moreover, this requires that:

$$s^{(1)} < v_1^* = v_0 + \frac{v_{n+1} - v_0}{n+1} \tag{5.40}$$

$$s^{(2)} > v_n^* = v_{n+1} - \frac{v_{n+1} - v_0}{n+1} \tag{5.41}$$

Now suppose that constraint c_2 is active and c_3 is not active. Then,

$$v_1^* = s^{(1)} \geq v_0 + \frac{v_{n+1} - v_0}{n+1} \tag{5.42}$$

where the inequality is just the negation of condition (5.40), and $\lambda_3^* = 0$. It follows from equations (5.38a) and (5.38c) that:

$$v_i^* = s^{(1)} + (i-1)\,\frac{v_{n+1} - s^{(1)}}{n}, \qquad i = 1, \dots, n \tag{5.43}$$

Furthermore, since constraint c_3 is not active:

$$s^{(2)} > v_n^* = v_{n+1} - \frac{v_{n+1} - s^{(1)}}{n} \tag{5.44}$$

which is a stricter condition than (5.41).

Similarly, supposing that constraint c_2 is not active and c_3 is active, if follows that:

$$v_n^* = s^{(2)} \leq v_{n+1} - \frac{v_{n+1} - v_0}{n+1} \tag{5.45}$$

$$v_i^* = v_0 + i\,\frac{s^{(2)} - v_0}{n}, \qquad i = 1, \ldots, n \tag{5.46}$$

and

$$s^{(1)} < v_1^* = v_0 + \frac{s^{(2)} - v_0}{n} \tag{5.47}$$

which is also stricter than condition (5.40).

Finally, both constraints are active in all other cases, i.e. whenever:

$$s^{(1)} \geq v_0 + \frac{s^{(2)} - v_0}{n} \tag{5.48}$$

$$s^{(2)} \leq v_{n+1} - \frac{v_{n+1} - s^{(1)}}{n} \tag{5.49}$$

and it follows directly from equation (5.38a) that:

$$v_i^* = s^{(1)} + (i-1)\,\frac{s^{(2)} - s^{(1)}}{n-1}, \qquad i = 1, \ldots, n \tag{5.50}$$

The following main result can now be stated:

Theorem 5.11 (Optimal μ-Distribution of the HSR Indicator). *Given two reference points $l, u \in \mathbb{R}^2$, $l \ll u$, a (bounded) linear nondominated front $\mathrm{X} = \{(v, f(v)) \mid s^{(1)} \leq v \leq s^{(2)})\} \subset [l, u]$, where $f(v) = -mv + b$, $m, b \in \mathbb{R}$, and $m > 0$, such that at least one point in X strongly dominates u, and a positive integer $\mu = n$, the optimal μ-distribution of the HSR indicator on X is the set $\{(v_i^*, f(v_i^*)) \mid i = 1, \ldots, n\} \subset \mathrm{X}$, where, letting $v_0 = (b - u_2)/m$ and $v_{n+1} = u_1$:*

$$v_i^* = \begin{cases} v_0 + i\,\frac{v_{n+1} - v_0}{n+1} & \textit{if } \; s^{(1)} < v_0 + \frac{v_{n+1}-v_0}{n+1} \; \wedge \; s^{(2)} > v_{n+1} - \frac{v_{n+1}-v_0}{n+1} \\ s^{(1)} + (i-1)\,\frac{v_{n+1} - s^{(1)}}{n} & \textit{if } \; s^{(1)} \geq v_0 + \frac{v_{n+1}-v_0}{n+1} \; \wedge \; s^{(2)} > v_{n+1} - \frac{v_{n+1}-s^{(1)}}{n} \\ v_0 + i\,\frac{s^{(2)} - v_0}{n} & \textit{if } \; s^{(1)} < v_0 + \frac{s^{(2)}-v_0}{n} \; \wedge \; s^{(2)} \leq v_{n+1} - \frac{v_{n+1}-v_0}{n+1} \\ s^{(1)} + (i-1)\,\frac{s^{(2)} - s^{(1)}}{n-1} & \textit{if } \; s^{(1)} \geq v_0 + \frac{s^{(2)}-v_0}{n} \; \wedge \; s^{(2)} \leq v_{n+1} - \frac{v_{n+1}-s^{(1)}}{n} \end{cases} \tag{5.51}$$

Moreover, the corresponding optimal investment is given by:

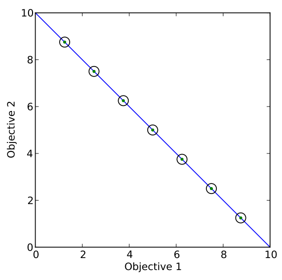
$$y_i^* = \frac{v_{i+1}^* - v_{i-1}^*}{w^*} \tag{5.52}$$

$$x_i^* = \frac{v_{i+1}^* - v_{i-1}^*}{v_{n+1} - v_0 + v_n^* - v_1^*} \tag{5.53}$$

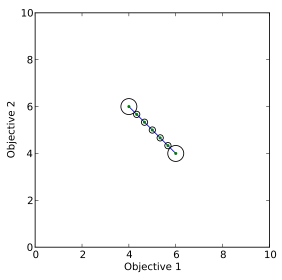
for $i = 1, \ldots, n$, where

$$w^* = \frac{m}{(u_2 - l_2)(u_1 - l_1)} \sum_{i=1}^{n} (v_{n+1} - v_i^*)\,(v_{i+1}^* - v_{i-1}^*)\,(v_i^* - v_0) \tag{5.54}$$

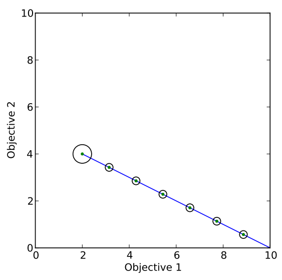
and, with a slight abuse of notation, $v_0^* = v_0$ and $v_{n+1}^* = v_{n+1}$.



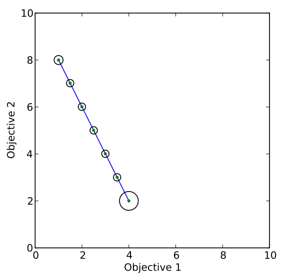
(a) $m = 1$, $b = 10$, $s^{(1)} = 0$, $s^{(2)} = 10$



(b) $m = 1$, $b = 10$, $s^{(1)} = 4$, $s^{(2)} = 6$



(c) $m = 1/2$, $b = 5$, $s^{(1)} = 2$, $s^{(2)} = 10$



(d) $m = 2$, $b = 10$, $s^{(1)} = 1$, $s^{(2)} = 4$

Figure 5.3: Optimal distribution and investment for $\mu = 7$ points on various linear fronts. Circle area is proportional to assigned investment. $l = (0, 0)$ and $u = (10, 10)$.

Proof. The above analysis already establishes that degenerate solutions cannot be globally optimal (Lemma 5.10) and that (y^*, v^*) is the only nondegenerate solution of Problem 5.2 that satisfies the KKT conditions. To show that it is indeed a minimal solution, note that the feasible region of Problem 5.2 is bounded above and below with respect to v, whereas it is bounded below with respect to y, but not necessarily bounded above (depending on $s^{(1)}$ and $s^{(2)}$). However, given that $y_i \geq 0$ for all $i = 1, \ldots, n$, the objective function $y^T P y$ is clearly bounded below by zero. Furthermore, $y^T P y$ tends to $+\infty$ whenever any component of y does so if the solution is not degenerate, and to a sub-optimal value if the solution is degenerate. Since it is also a continuous function, it must have a (global) minimum, and the corresponding nondegenerate feasible solution must be finite. Given that there is only one nondegenerate solution that satisfies the KKT conditions, such a solution must be both a locally and globally minimal solution. □

Examples

Figure 5.3 shows the optimal μ-distributions considering $\mu = 7$ points and their corresponding optimal investment for linear fronts with different slopes and different domain boundaries. The blue solid lines represent the fronts. The area of the circle around each point is proportional to the investment assigned to it.

It can be seen that all points are equally spaced, and that all inner points are assigned equal investment, whereas the outer points may be assigned greater or equal investment than the inner points depending on the location of the front end points. In particular, front end points that are sufficiently far away from the corresponding

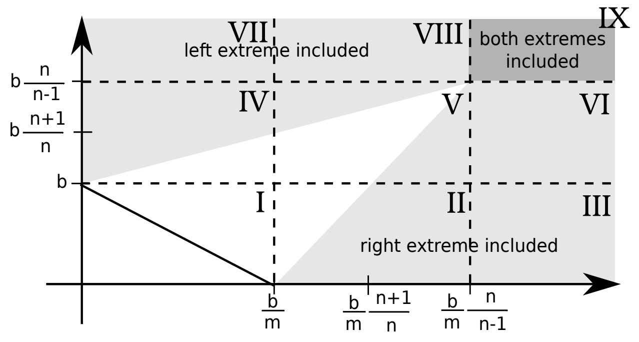


Figure 5.4: Illustration of which extreme points, if any, are included in the optimal μ-distribution for a linear front defined by $f(v) = -mv + b$, $v \in \left[0, \frac{b}{m}\right]$, depending on the location of the reference point. After [40].

upper boundaries of the region of interest receive greater investment than the inner points.

5.4.4 Comparison with the Hypervolume Indicator

The results obtained for the optimal μ-distribution of the HSR indicator on two-objective linear fronts are consistent with those reported in the literature for the hypervolume indicator [40]. In particular, all points are uniformly spaced on the front, and the outer points may or may not coincide with the front's extreme points. It remains to show whether the outer points are exactly the same for both indicators (given an appropriate setting of the reference points, l and u).

For the HSR indicator, the outer points of the optimal μ-distribution on a linear front depend only on how $s^{(1)}$, $s^{(2)}$, v_0 and v_{n+1} relate to one another, see the branch conditions in expression (5.51). This implicitly answers the question of whether the front's extreme points, $(s^{(1)}, f(s^{(1)}))$ and $(s^{(2)}, f(s^{(2)}))$, are included in the optimal μ-distribution or not. In contrast, for the hypervolume indicator, this issue was studied from the point of view of where to place the reference point in order to include none, just one, or both extreme points of the linear front [40]. The connection between the results on the outer points of the optimal μ-distributions of the two indicators is not evident, and deserves further attention.

Brockhoff [40] showed how the inclusion of one or both extreme points in the optimal μ-distribution of the hypervolume indicator on a given linear front $f(x) = -mx+b$, $m, b \in \mathbb{R}^+$, with bounded domain $\left[0, \frac{b}{m}\right]$, without loss of generality, depends on the position of the reference point, which he denoted by r (not to be confused with the return vector r in Subsection 2.4.5).

Brockhoff concluded that, as illustrated in Figure 5.4, if the reference point is weakly dominated by the point $\left(\frac{n}{n-1}\frac{b}{m}, \frac{n}{n-1}b\right)$ (case IX), then both extremes are included in the optimal μ-distribution. Otherwise, the left extreme point is included only if $r_2 \geq \frac{n+1}{n}b - \frac{f(r_1)}{n}$, and the right extreme point is included only if $r_1 \geq \frac{n+1}{n}\frac{b}{m} - \frac{f^{-1}(r_2)}{n}$. If none of the above conditions are satisfied, then none of the extreme points are included in the optimal μ-distribution. In particular, this is obvious for $r \leq \left(\frac{b}{m}, b\right)$. Moreover, when the left extreme point is not included, the left outer point is $v_1 = \frac{v_2+v_0}{2}$, and when the right extreme point is not included, the right outer point is $v_n = \frac{v_{n+1}+v_{n-1}}{2}$, where $v_{n+1} = r_1$ and $v_0 = f^{-1}(r_2)$ as defined previously. This is deduced from equation (8) in [40], by noting that $F_l = r_2$ when the left extreme point is not included, and $F_r = r_1$ when the right extreme point is not included.

With an appropriate setting of the reference points l and u, the outer points of the optimal μ-distribution of the two indicators can be shown to be exactly the same. Assume, as in Brockhoff's study [40], that the linear front is given by $f(x) = -mx+b$, where $m, b \in \mathbb{R}^+$, and has bounded domain $\left[0, \frac{b}{m}\right]$. Then, the left and right extreme points are $(0, b)$ and $\left(\frac{b}{m}, 0\right)$, respectively. Recall that, by Theorems 5.1 and 5.2, the exact location of the lower reference point, l, does not affect the optimal investment as long as l weakly dominates every point in the front, and, by Theorem 5.11, neither does it affect the optimal μ-distribution of the HSR indicator on linear fronts. Thus, it suffices to assume that $l \leq (0, 0)$.

Setting $u = r$, and as with the hypervolume indicator, if the left extreme point is not included in the optimal μ-distribution of the HSR indicator then $v_1 = \frac{v_2+v_0}{2}$ (branches 1 and 3 of equation (5.51)), and if the right extreme point is not included, then $v_n = \frac{v_{n+1}+v_{n-1}}{2}$ (branches 1 and 2 of equation (5.51)). It is therefore sufficient to show that the conditions under which each extreme point is included in the optimal μ-distribution are the same for both the HSR and the hypervolume indicators.

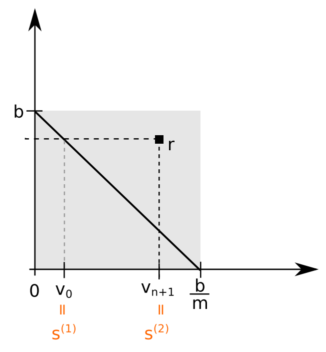
In the following analysis, it is always assumed that the (upper) reference point $u = r$ is such that it is strongly dominated by at least one point in the bounded linear front. To match the conditions of Theorem 5.11, which requires the front to be contained in the region of interest $[l, u]$, any part of the front that does not weakly dominate u is discarded by adjusting the end points of the domain to $s^{(1)} = \max(0, v_0)$ and $s^{(2)} = \min\left(\frac{b}{m}, v_{n+1}\right)$, where $v_0 = f^{-1}(r_2)$ and $v_{n+1} = r_1$. This is consistent with the fact that such points have zero return and, therefore, cannot be included in the optimal μ-distribution anyway.

The analysis is split into four scenarios, each relating to one of the four possible locations of the reference point r, represented in terms of v_0 and v_{n+1}, relatively to the point $\left(\frac{b}{m}, b\right)$. Each of these scenarios is then linked to the corresponding case(s) from I to IX in Figure 5.4 that reflect the results for the hypervolume indicator. Figure 5.5 illustrates the four scenarios in more detail, and establishes the connection between Figure 5.4 and the notation used in Section 5.4.3.

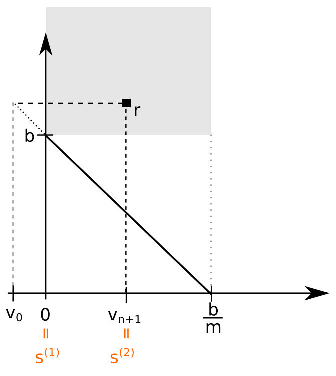
The scenarios are the following:

1. **$v_0 > 0$ and $v_{n+1} < \frac{b}{m}$ (case I)**
   In this scenario, $s^{(1)} = v_0$ and $s^{(2)} = v_{n+1}$ (see Figure 5.5(a)). It is clear that the first branch of expression (5.51) is satisfied in this case. This indicates that, when $r \leq \left(\frac{b}{m}, b\right)$, none of the extreme points with x-coordinate $s^{(1)}$ and $s^{(2)}$ are included (and neither are the original extreme points, $(0, b)$ and $(\frac{b}{m}, 0)$). Furthermore, points are equally spaced between v_0 and v_{n+1}, as for the hypervolume indicator.

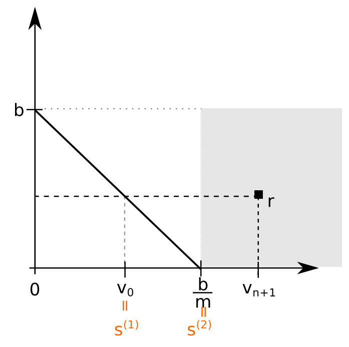
2. **$v_0 \leq 0$ and $v_{n+1} < \frac{b}{m}$ (cases IV and VII)**
   In this scenario, $s^{(1)} = 0$ and $s^{(2)} = v_{n+1}$ (see Figure 5.5(b)). Since $s^{(2)} = v_{n+1}$, only the conditions on $s^{(2)}$ of branches 1 and 2 of expression (5.51) are satisfied, and, therefore, the right extreme point is not included. By manipulating the condition on $s^{(1)}$ of (one of) those branches while taking into account that $s^{(1)} = 0$, $f(x) = -mx + b$, and $f(v_0) = r_2$, it is possible to conclude that the



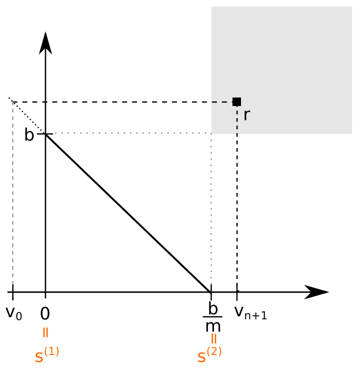
(a) case I



(b) cases IV and VII



(c) cases II and III



(d) cases V, VI, VIII and IX

Figure 5.5: Possible locations of the reference point, r, relatively to the point $\left(\frac{b}{m}, b\right)$. The grey area indicates the region considered in each case.

left extreme point is included (i.e., branch 2 applies) only if $r_2 \geq \frac{n+1}{n}b - \frac{f(r_1)}{n}$.

3. **$v_0 > 0$ and $v_{n+1} \geq \frac{b}{m}$ (cases II and III)**
In this scenario, $s^{(1)} = v_0$ and $s^{(2)} = \frac{b}{m} \leq v_{n+1}$ (see Figure 5.5(c)). Since $s^{(1)} = v_0$, only the conditions on $s^{(1)}$ of branches 1 and 3 of expression (5.51) hold and, therefore, the left extreme point is not included. By manipulating the condition on $s^{(2)}$ of (one of) those branches while taking into account that $s^{(2)} = \frac{b}{m}$, $f(x) = -mx + b$, $v_0 = f^{-1}(r_2)$, and $v_{n+1} = r_1$, it is possible to conclude that the right extreme point is included (i.e., branch 3 applies) only if $r_1 \geq \frac{n+1}{n}\frac{b}{m} - \frac{f^{-1}(r_2)}{n}$.

4. **$v_0 \leq 0$ and $v_{n+1} \geq \frac{b}{m}$ (case V, VI, VIII and IX)**
In this last scenario, $s^{(1)} = 0$ and $s^{(2)} = \frac{b}{m}$ (see Figure 5.5(d)). To see whether or not the extreme points are included, the conditions on r_1 and r_2 for which the fourth branch in expression (5.51) applies, which corresponds to case IX, are derived first. Then, the remaining cases V, VI and VIII are studied by negating either or both of those conditions.

   (a) **(case IX)** Taking into account that $v_0 = f^{-1}(r_2)$ and $v_{n+1} = r_1$, the conditions on $s^{(1)}$ and $s^{(2)}$ of the fourth branch in (5.51) can be manipulated in order to get to the equivalent conditions $r_2 \geq \frac{n}{n-1}b$ and $r_1 \geq \frac{n}{n-1}\frac{b}{m}$, respectively. If r is such that both conditions are met then both extreme points are included.

(b) **(case VI)** If $r_1 \geq \frac{n}{n-1}\frac{b}{m}$ and $r_2 < \frac{n}{n-1}b$, then $b \leq r_2 < \frac{n}{n-1}b$, and the condition on $s^{(1)}$ in the fourth branch of expression (5.51) is not satisfied. However, the condition on $s^{(2)}$ in branch 3 can be shown, by contradiction, to be satisfied for r within the specified range. Therefore, branch 3 applies, and only the right extreme point is included.

(c) **(case VIII)** If $r_1 < \frac{n}{n-1}\frac{b}{m}$ and $r_2 \geq \frac{n}{n-1}b$, then $\frac{b}{m} \leq r_1 < \frac{n}{n-1}\frac{b}{m}$, and the condition on $s^{(2)}$ in the fourth branch of expression (5.51) is not satisfied. However, the condition on $s^{(1)}$ in branch 2 can be shown, by contradiction, to be satisfied for r within the specified range. Therefore, branch 2 applies, and only the left extreme point is included.

(d) **(case V)** If neither $r_2 \geq \frac{n}{n-1}b$ nor $r_1 \geq \frac{n}{n-1}\frac{b}{m}$ are satisfied, then it follows that $\frac{b}{m} \leq r_1 < \frac{n}{n-1}\frac{b}{m}$ and $b \leq r_2 < \frac{n}{n-1}b$. By manipulating the condition on $s^{(1)}$ in branch 2 as done before for cases IV and IIV, it follows that, if $r_2 \geq \frac{n+1}{n}b - \frac{f(r_1)}{n}$, branch 2 applies and the left extreme point is included. Analogously, by manipulating the condition on $s^{(2)}$ in branch 3 as done before for cases II and III, it follows that, if $r_1 \geq \frac{n+1}{n}\frac{b}{m} - \frac{f^{-1}(r_2)}{n}$, then branch 3 applies and the right extreme point is included. Finally, if neither neither the condition on $s^{(1)}$ in the second branch nor the condition on $s^{(2)}$ in the third branch are satisfied, then both conditions in the first branch of expression (5.51) must be satisfied, and none of the extreme points are included in the optimal μ-distribution.

Summing up, the optimal μ-distribution for the HSR indicator on a linear front is exactly the same as for the hypervolume indicator.

5.5 Experimental Results

The aim of this section is to provide insight into the optimal μ-distributions and corresponding optimal investment for the HSR indicator on other than linear 2-dimensional fronts, through numerical approximations. The influence of the reference points is investigated as well. Firstly, the optimal investment is investigated in Section 5.5.1, by observing how it is distributed among random and evenly distributed point sets on several fronts (Figures 5.6 and 5.7). Secondly, in Section 5.5.2, the optimal μ-distributions and corresponding optimal investments are approximated (Figures 5.9 and 5.10) and compared to analogous approximations for the hypervolume indicator.

5.5.1 Optimal Investment

Figures 5.6 and 5.7 show the optimal investment assigned by HSR indicator on 20 points randomly and evenly distributed, respectively, on different fronts (columns), and for different settings of the reference points l and u (rows). The plots depict the orthogonal space $[l, u]$ after normalization. The first row represents the base line, where the front domain is $[0, 1]$ and the reference points are $l = (0, 0)$ and $u = (1, 1)$. The second to fourth rows correspond to shifting either one or the two reference points away from the front and along a straight line containing the points $(0, 0)$

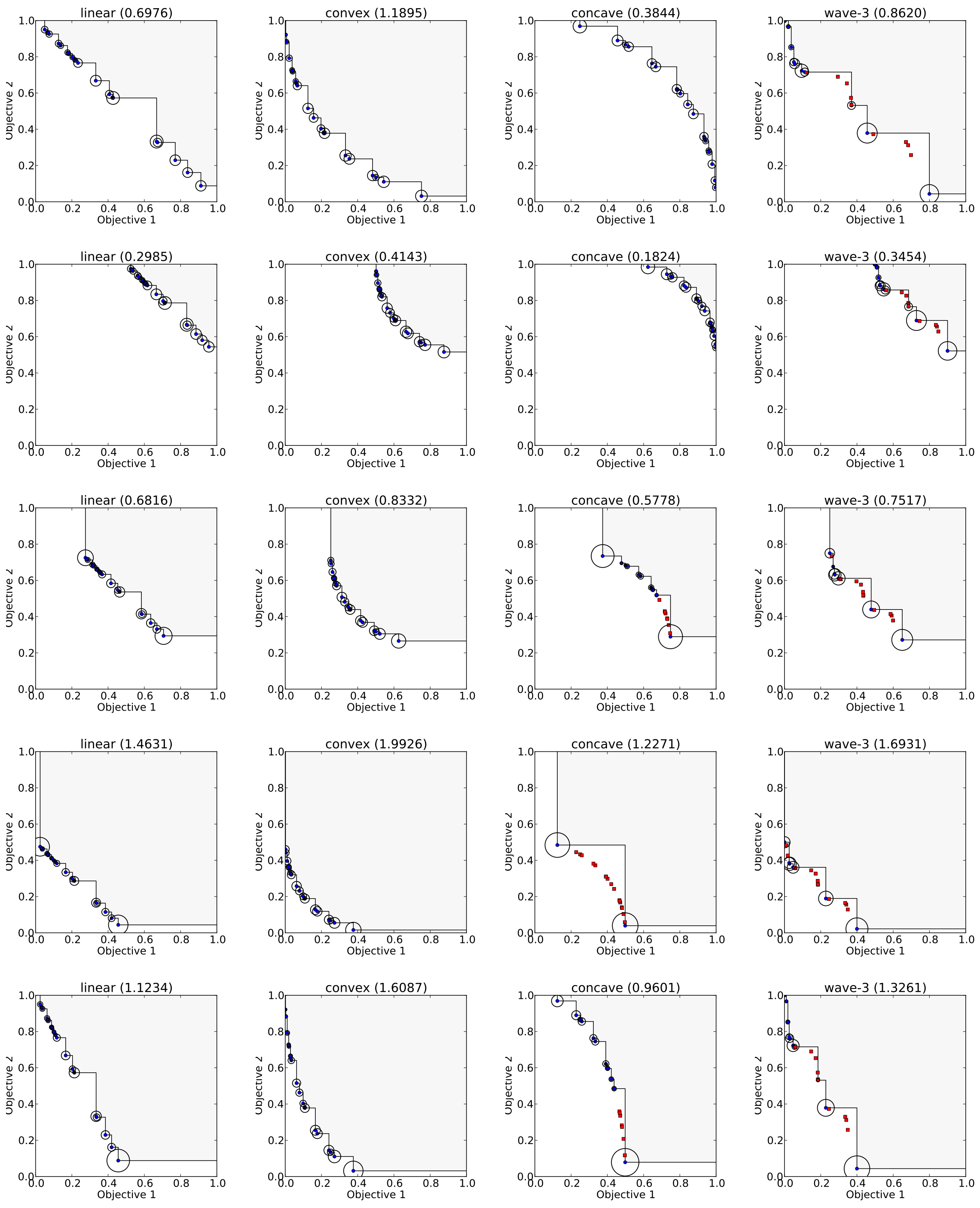


Figure 5.6: Examples of the optimal investment of HSR indicator on 20 points randomly distributed along linear, convex, concave and wave-3 fronts, respectively from left to right. The first row shows fronts in the $[0, 1]$ domain where $l = (0, 0)$ and $u = (1, 1)$. The second to fourth rows correspond to the front in the first row but with l and u respectively set as follows: $l = (-1, -1)$ and $u = (1, 1)$, $l = (-0.5, -0.5)$ and $u = (1.5, 1.5)$, $l = (0, 0)$ and $u = (2, 2)$, and $l = (0, 0)$ and $u = (2, 1)$. The plots show the $[l, u]$ space after normalization. The region dominated by the points with strictly positive investment is filled gray. The Sharpe ratio assigned to each point set is shown in the title.

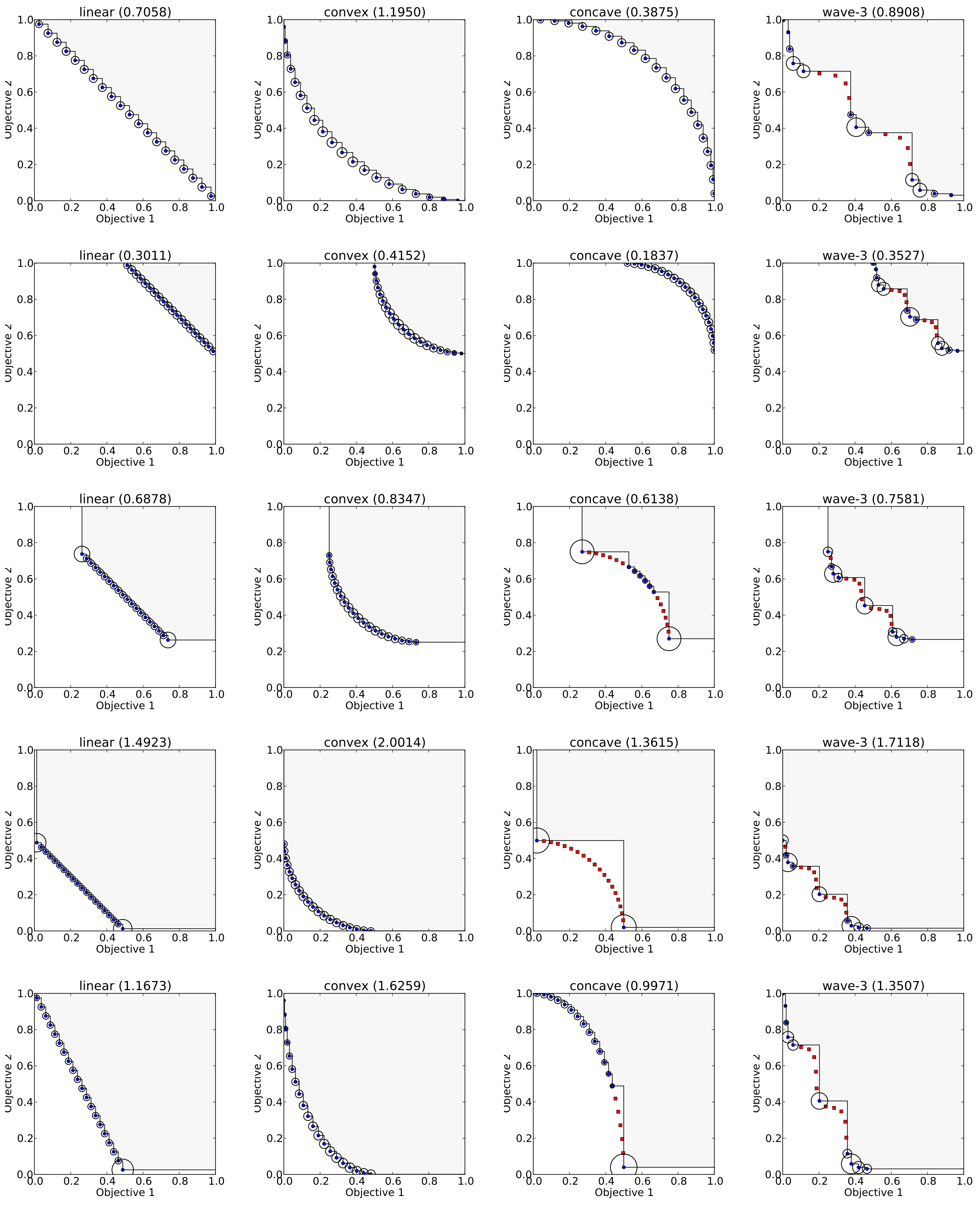


Figure 5.7: Examples of fronts and settings of l and u analogous to those in Figure 5.6 where instead of starting with points randomly distributed on the $[0, 1]$ domain, they are evenly distributed.

and $(1, 1)$. The settings of l and u are such that the distance between every pair of points is kept the same for the three settings. In practice, such settings correspond to shrinking the front in the first row and translating them along the diagonal going through points $(0, 0)$ and $u = (1, 1)$. Unlike in the other rows, the settings of l and u in the last row do not preserve the proportion between axes. Note that, in Figure 5.7, moving u only along the y-axis (as in the last row) is symmetrical to moving it only along the x-axis. Moreover, as proved in Section 5.4.1, the plots in the second row show that changing l does not affect the optimal investment.

Each point is represented in the plots either as a blue circle or a red square, representing whether the point was assigned an investment higher or lower than $1e^{-16}$, respectively. The blue points will be the center of a circle whose area is proportional to the investment assigned to that point. Therefore, the higher the area of a circle around a point is, the higher was the investment in that point.

In the case of linear fronts (first column of both figures), as expected from the theoretical results, the investment in a point is directly related to distance between its two neighbors. The further apart they are from each other, the higher is the investment in the point. If points are evenly distributed, then so is the investment. The investment in extreme points depends on the upper reference point, u. Moving u away increases the investment in the extreme points. As expected, all (nondominated) points in a linear front have a strictly positive investment.

In the case of convex fronts (second column of both figures), all points are also assigned strictly positive investment. It seems that, as in the case of the linear front, the investment in the extreme points increases as the upper reference point u moves away from the front, although it does not increase as much. The scale distortion (last row) seems to have an effect on the convex front different from the effect on the linear front. In this case, there is a shift of investment from the side of the front further away from l in the y-coordinate to the side of the front (and the points) closer to l (this can be observed comparing the first and the last rows). This seems to happen so that more investment is assigned to points with higher expected return.

On concave fronts (third column), the observations are not exactly the same. The experiments show that some nondominated points are assigned zero investment or at least, it is very close to zero. Consider that the direct neighbors of a point are the closest points on each side that have strictly positive investment. In that case, the (positive) investment in a point seems to be related to the distance between its direct neighbors and to how much investment is assigned to them. The investment in a point also seems to depend on how far from u it is. The points in the concave region have positive investment if the front is close enough to u, but as it moves towards l, investment decreases and, if far enough apart from u, investment goes to zero. In that case, the investment in the inner points (points in the concave region) is transferred to the extreme points of the front. In the fifth row of both figures, more points on one side of the front have positive investment. The right extreme receives most of the investment, which is much more than the left extreme receives. This may be because the right extreme point has the highest expected return of all points. The remaining investment is assigned to points in left side. It seems that the other points on the right side of the front receive (almost) no investment because of how much is assigned to the right extreme point.

The wave-3 front that has both concave and convex regions provides a better insight into the optimal investment on arbitrary fronts. It is observable that preference is given to extreme points, and to points in convex regions (knees) over points in concave regions. The preference for convex regions is also evident in Figure 5.8, which shows how changing the position of a point inside the region bounded by the two closest points to it affects the optimal investment, considering points evenly distributed on different fronts. It can be observed that moving a point closer to l results in assigning more investment to it and less to its closest neighboring points. On the other hand, as the point is moved closer to u, the investment it gets decreases, and more is assigned to its closest neighbors.

Comparing Figures 5.6 and 5.7, sets of points evenly distributed were preferred over the randomly distributed ones, i.e., they all had higher Sharpe ratios. On linear fronts, that is a characteristic of the optimal μ-distribution on such fronts. However, Figure 5.7 indicates that such an observation does not generalize to other fronts, i.e., the optimal μ-distribution on an arbitrary front does not consist of a set of uniformly distributed points. The results indicate that, optimal μ-distributions are such that points are concentrated on the extremes and on convex regions.

5.5.2 Approximating Optimal μ-Distributions

The exact optimal μ-distributions and corresponding optimal investment for the HSR indicator are now known for two-dimensional linear fronts. As extending such results to other fronts is not trivial, the alternative is to empirically approximate the optimal μ-distributions. This section shows numerical approximation results for different fronts and for different settings of the reference points. Moreover, such optimal μ-distribution approximations are compared to those for the hypervolume indicator.

Experimental Setup

To approximate the optimal μ-distribution of an indicator for a given front, a non-linear solver from the *scipy* python module was used (scipy.optimize.fmin_slsqp). A vector, the bounds for each variable, and a function to be optimized are passed to the solver. The μ-sized vector represents the x-coordinates of μ points in the front, and the bounds represent the corresponding domain ranges.

As the considered fronts are continuous and differentiable, the y-coordinates can be determined from the x-coordinates. Therefore, the function receives the vector, determines the corresponding y-coordinates (depending on the front being considered) and then, computes and returns the indicator value. The solver is used to approximate the vector of x-coordinates that minimizes the negative of the function, i.e., to approximate the optimal μ-distribution that maximizes an indicator for the front under consideration. Two indicators were considered, the HSR indicator and the hypervolume indicator. In the former case, the function to be optimized receives the vector and after computing the corresponding expected returns and the covariance matrix, computes the optimal investment using a QP solver and returns the corresponding Sharpe ratio. In the latter case, the hypervolume indicator is computed and returned.

The results presented show, for each front and setting of the reference points, the best approximation for $\mu = 10$ obtained after 30 repetitions with different random starting vectors. The first experiments were ran for 2-dimensional continuous and differentiable fronts restricted to the $[0, 1] \times [0, 1]$ region, similar to those used in

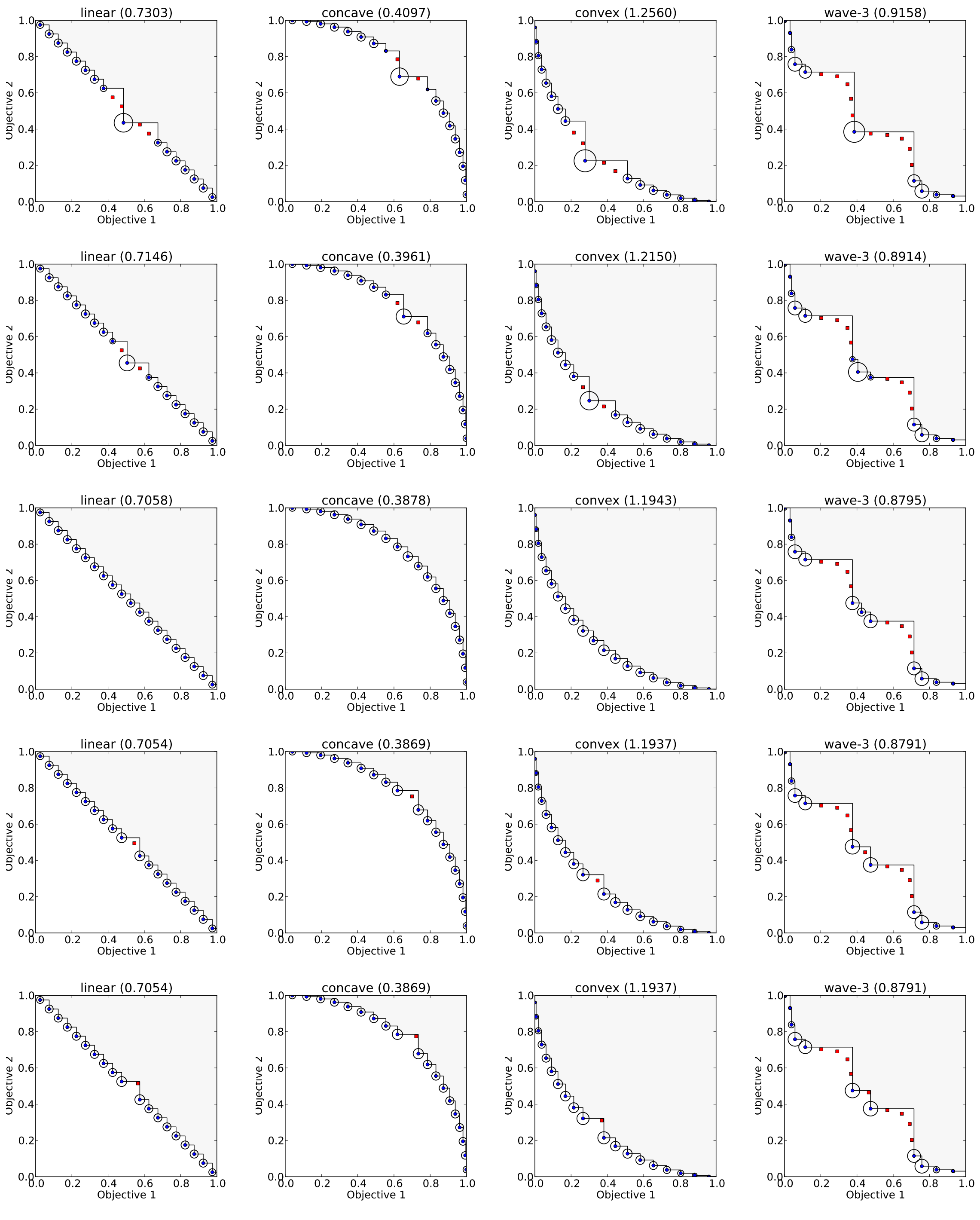


Figure 5.8: Example of the change in investment with the change of location of a single point in different fronts.

Section 5.5.1. The fronts considered were the concave, convex and the wave-w (where $w \in \{1, 3\}$) fronts (see Section 3.5). The remaining experiments were made on fronts that result from applying the same transformation as in Section 5.5.1 through different settings of the reference points and normalizing the space afterwards.

Results

Figures 5.9 and 5.10 show the approximated optimal μ-distributions for the HSR indicator (odd columns) and for the hypervolume indicator (even columns). The settings of l and u for each row are the same as those for Figures 5.6 and 5.7. In this case, Figure 5.9 shows the results for the convex and the concave fronts and Figure 5.10 shows the results for the wave-1 and wave-3 fronts. The plots also show the Sharpe ratio and hypervolume values for each distribution. Always recall that these are just approximations and therefore, the conclusions have to be carefully drawn. For example, note that, for the concave fronts in the forth row of Figure 5.9, the approximated distribution for the hypervolume indicator leads to a better Sharpe ratio than that found for the HSR indicator.

The first conclusion is that, in general, a higher Sharpe ratio does not imply a higher hypervolume, and vice-versa. Compare, for example, the first two plots of the first row of Figure 5.9 and notice that the first one has higher Sharpe ratio but lower hypervolume indicator than the second plot. This and the results in the figures indicate that, in contrast to linear fronts, the optimal μ-distributions of HSR and hypervolume indicators are not the same, in general. In particular, this is clearly suggested by the results on the concave and wave fronts.

Concerning the convex front, both approximations are similar. The μ-distribution for the HSR indicator seems to be such that points are more or less evenly distributed, but slightly more closer to one another than that for the hypervolume indicator. In the case of the concave front, points are more evenly spaced in the optimal μ-distributions for the hypervolume indicator. In both cases, the approximations indicate that, when the front is close enough to u, the optimal μ-distributions tend to be almost the same for the two indicators. Moreover, as the front is farther away from the upper reference point, the results indicate that the distance between the outer points and their single neighbor increases more than the distance between an inner point and its inner-point neighbors. This seems to be more evident in the case of HSR indicator, to the point that it may not even be worthwhile to invest in inner points if the front is far away enough from u. Hence, it may be the case that, for some fronts and some settings of u, there is an optimal μ-distribution for the HSR indicator that has less than μ points, which does not happen in the case of the hypervolume indicator. However, as mentioned before, these are just approximations and, therefore, this conclusion is not certain, particularly because of the (clearly sub optimal) results in the fourth row of Figure 5.9.

In the case of the wave-w ($w \in \{1, 3\}$), the results in Figure 5.10 indicate that, for the HSR indicator, points are preferably concentrated in convex regions, more so than in the case of the hypervolume indicator, and fewer points are placed in concave regions. Once again, the results indicate that the optimal μ-distribution of the HSR indicator may contain fewer than μ points, as the results show some points with (very close to) zero investment.

5.6 Extended Hypervolume Sharpe-Ratio Indicator

The model on which the HSR indicator is based considers the uncertainty of a single Decision Maker regarding the quality of solutions that are mapped onto a

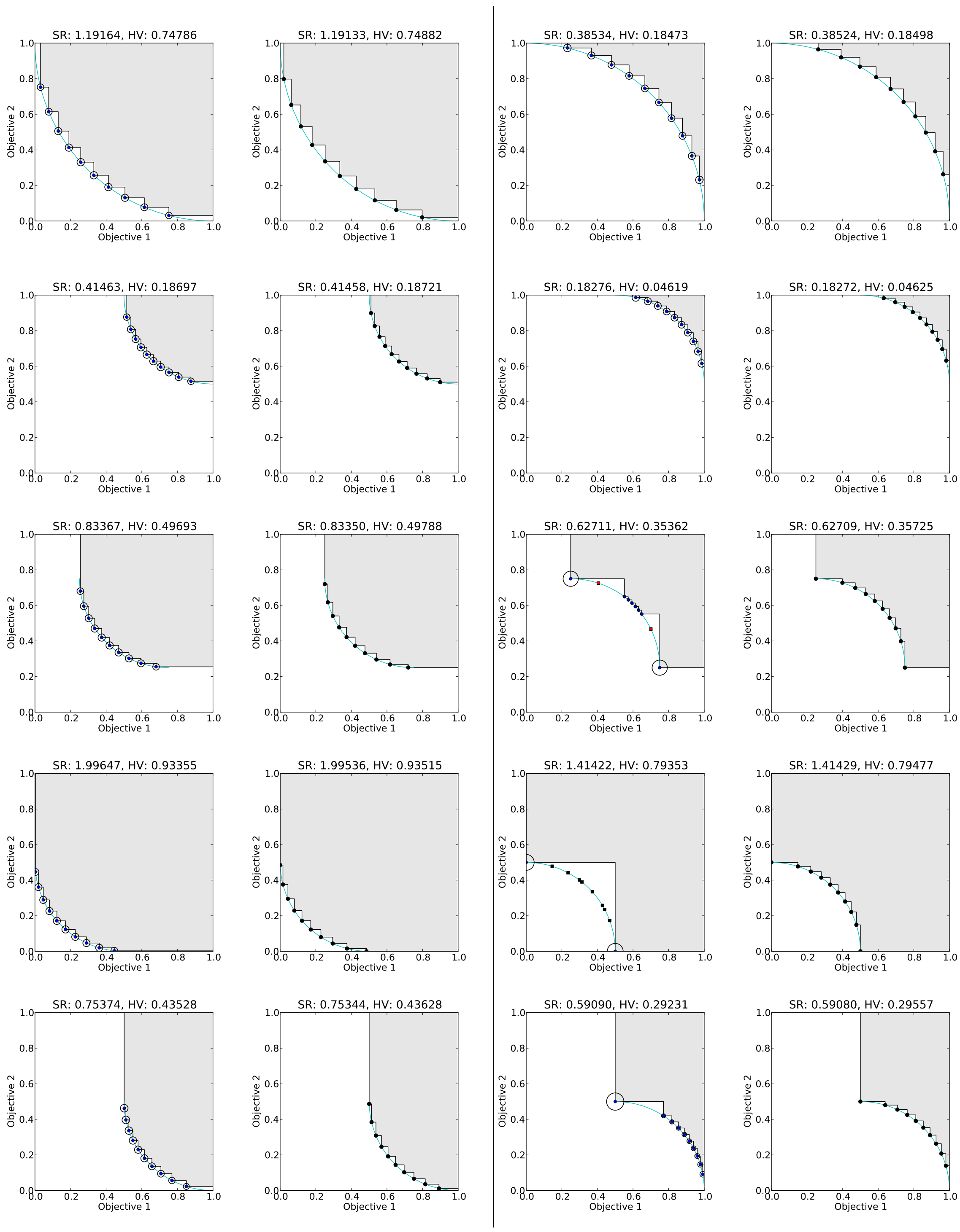


Figure 5.9: Examples of the approximation of optimal μ-distributions for the HSR indicator (odd columns) and corresponding optimal investment and for the hypervolume indicator (even columns) for the convex front (first two columns) and the concave front (last two columns). Each row represents a different setting for l and u, the same used in Figures 5.6 and 5.7. The Sharpe ratio (SR) and the hypervolume indicator (HV) of each point set is shown in the plot title.

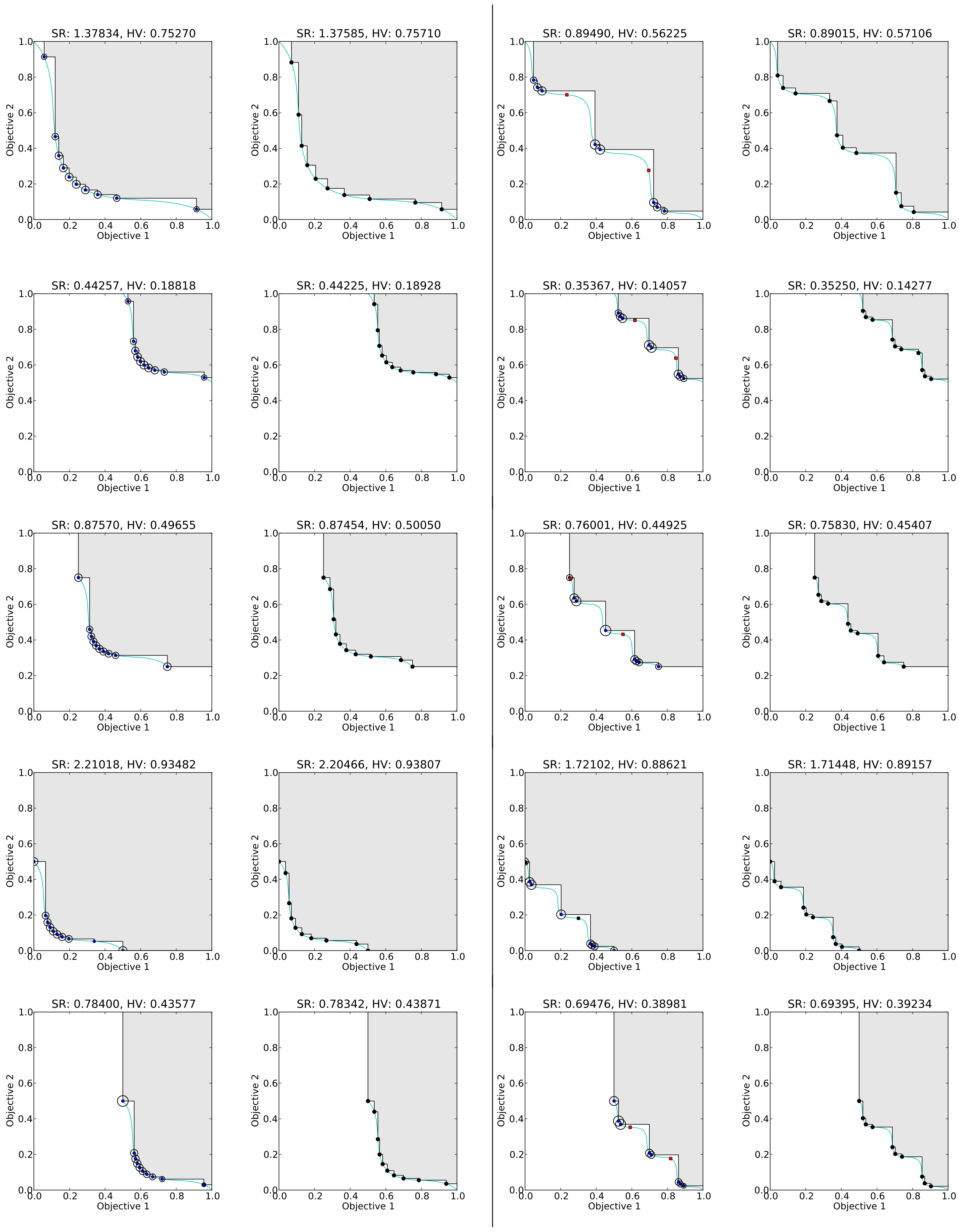


Figure 5.10: Examples of the approximation of optimal μ-distributions for the HSR indicator (odd columns) and corresponding optimal investment and for the hypervolume indicator (even columns) for the wave-1 front (first two columns) and the wave-3 front (last two columns). Each row represents a different setting for l and u, the same used in Figures 5.6 and 5.7. The Sharpe ratio (SR) and the hypervolume indicator (HV) of each point set is shown in the plot title.

unique point in objective space. However, several problems benefit from a problem formulation where each solution in the decision space is mapped onto a set of (nondominated) points in the objective space. The HSR indicator can be easily extended to handle such a formulation.

In this section, problems with set-valued solutions are explained in Section 5.6.1 using application examples. The additional difficulties raised in evaluating such solutions in comparison to solutions of (unconstrained) multiobjective optimization problems are discussed also in Section 5.6.1. The uncertainty model for the problems with set-based solutions is formulated in Section 5.6.2, and experiments on the resulting instance of the Sharpe-ratio class are shown and discussed in Section 5.6.3.

5.6.1 Motivation

In multiobjective optimization, a solution in decision space is typically mapped onto a single point in objective space. However, for some problems, a mapping onto a set of (nondominated) points in objective space may provide additional insight to the DM, and allow a reduction on the search space and/or on the dimensionality of the objective space. Such problems will be referred to here as Set-Valued Optimization (SVO) Problems. Figure 5.11 shows an example of the mapping between decision space and objective space for this type of problems. Such a mapping is expressed as a function $f : \Omega \to 2^{\mathbb{R}^d}$, where Ω and $\mathbb{R}^d$ represent the decision and the objective spaces, respectively, and $2^{\mathbb{R}^d}$ denotes the power set of $\mathbb{R}^d$. In set-valued optimization with set-optimization criteria [1], a solution $x \in \Omega$ is a minimizer of f iff for all $x' \in \Omega$ such that $f(x') \preceq f(x)$ then $f(x) \preceq f(x')$. In the problems explored in this section, f maps onto a finite point set of which only the nondominated points are considered to be of relevance. Therefore, let us restrict to the case where f maps onto a discrete and finite nondominated point set. Problems to which the above formulation applies include: problems where solutions are evaluated over multiple points in time for which objective values increase/decrease monotonically (e.g., the electrical network restoration problem [45]); problems where solutions have to be evaluated under the same objectives considering multiple scenarios (e.g., multiobjective games [7], worst case robustness [6]); the problem of optimizing a limited-size set of solutions to a MO problem (e.g., population of-sets EAs [9]). In such problems, each solution has to be evaluated considering the set which it is mapped onto.

Electrical Network Restoration Problem

The electrical network restoration problem [45] is based on the need to re-configure an electrical network after one or more failures occur, e.g., due to adverse weather conditions, causing some parts of the network to become disconnected and, thus left without electricity. To restore the power in the network, a sequence of maneuvers has to be performed, where each maneuver consists in the activation/deactivation of a link between nodes in the network. These are performed in stages, for example, a stage may consist of a maneuver to deactive a link and another to activate another link. In general, the goal is to find which maneuvers to perform (which are grouped in stages) and in which order to perform the stages. By grouping maneuvers in stages, candidate solutions can be constructed in a way that no part of the network with electricity becomes disconnected after each stage. The electrical network should be reconfigured in such a way that the load is recovered as quickly as possible and the

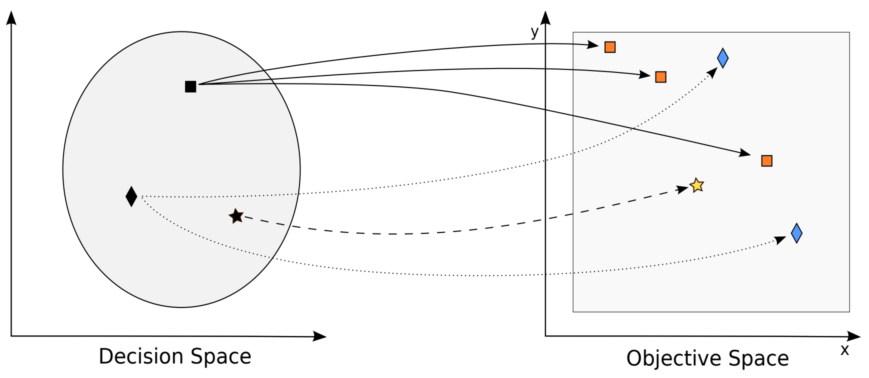


Figure 5.11: Example of 3 solutions in the decision space and the corresponding point sets in the objective space (identified by arrows of the same type and points of the same shape and color) of a set-valued optimization problem.

amount of unrecovered load is minimized, while satisfying all problem constraints at all steps.

Although the ideal solution would recover the full load in the shortest amount of time, it is also important for the DM to understand how much time is required and how much load is restored after every stage. This allows, for example, to help avoid having to pay fines to each client when the load is not established soon enough. Thus, the DM is interested not only in the evaluation of the solution after all stages are performed, but also after each intermediate stage. For example, suppose that there are two stages, A and B, that can be performed in any order but where A recovers 70% while B recovers the remaining 30% of the unattended load, and that both require one hour to be performed. Although the load is fully restored after performing the two stages, it is more advantageous for the DM to perform stage A first because more load will be restored sooner.

Carrano *et al.* [45] proposed to map every solution in the decision space into a nondominated point set in a two-dimensional objective space. They considered the minimization of the accumulated time required by the maneuvers/stages (objective 1) and the minimization of the unrecovered load (objective 2). In such a case, the set of points in objective space corresponding to the evaluation of a solution represents the evolution of the objective values as the stages are performed. In the above example, the solution where stage B is performed after A, would be mapped into the following two-point set $\{(1, 30), (2, 0)\}$. One of the benefits of this model is the reduction of the search space, as there is no need to consider solutions encoded by another solution, i.e., consisting of only the first stages of another solution.

Other Problems

Multiobjective Games [7] are another example where mapping each solution onto a point set may provide additional insight and possibly lead to better decision making. Each player's plan of action carries payoffs to each player (e.g., in checkers, it may be related to the number of pieces lost by each one). In such problems, the players have d objectives to optimize which are related to the payoffs at the end of the game (e.g., each player wishes to maximize the number of pieces lost by the opponent and minimize its own). Each player has several strategies that she/he may follow, where each one defines a course of action for the whole game (e.g., move the piece that is closer to the opponent's side). To optimize its own payoffs, a player has to choose a strategy, taking into account all strategies that the opponent may follow. Therefore, there are multiple payoff vectors associated to each player's strategy (solution), one for each of the opponent's possible strategies. From the point-of-view of one of the players, a worst-case scenario is considered, where for each of her/his strategies only the most beneficial strategies for the opponent are considered. Thus, each solution maps to a nondominated point set in the objective space.

Worst case robustness problems [6] consider that solutions are subject to uncertainties and thus, several scenarios have to be considered when evaluating them. In such a case, each solution is mapped to a set of points in the objective space, one point for each scenario considered. The goal is to find the best and most reliable/robust solutions to guarantee, as much as possible, the satisfaction of the problem constraints under the problem-related uncertainties. This is achieved by optimizing the worst case, which means that only the scenarios for which the solution performs the worse have to be considered. In practice, it translates to considering only the nondominated points each solution maps to. In such problems, the goal is to minimize the worst case, and preference is given to solutions whose worst-case performance map to a set of points with reduced spread (related to less uncertainty).

In population-based EMOAs working on sets of solutions, the goal is to find the "best" subset of optimal solutions (e.g., regarding a quality indicator) for a given multiobjective problem, where each individual represents a candidate subset. Therefore, such EMOAs aim at solving set-valued optimization problems. These and other EA methods for solving different SVO problems require individuals to be evaluated based on the corresponding point sets, and environmental selection on such individuals translates to performing subset selection on sets of point sets.

Evaluating Solutions

The evaluation of set-valued solutions adds another level of difficulty to the assessment of solutions and to the assessment of sets of solutions in comparison to the traditional formulation of a multiobjective optimization problem. The same questions apply: What is a good solution? What is a good (diverse) set of solutions? It is still clear that a (set-)dominated solution should not be preferred over a dominating one while nondominated solutions cannot be easily discarded without additional information regarding DM preferences. Under no preference information, although a (subjective) notion of diversity, related to the spread of solutions, is commonly accepted in the (single-point) multiobjective optimization case, it does not easily extend to set-valued optimization problems. That is, should it be related to having each solution covering one specific region of the objective space? Or each solution covering the objective space as much as possible? Or having solutions mapping to different set sizes?

The assessment/ranking and selection of solutions in EAs highly influences its capability to approach optimal solutions and to present the DM with a diverse set of good solutions. Although fitness assignment and selection in EAs for SVO problems becomes more difficult, it is important to understand the inner preferences imposed by the selection and/or fitness assignment method used, in order to understand its guarantees and the (possible) limitations imposed. It would be ideal to have an indicator with good properties such as monotonicity with respect to a set-dominance relation and which could be studied regarding (set-based) μ-distributions.

As highlighted by Carrano *et al.* [45], most of the well-known EMOAs are not easily adapted to carry out the optimization considering the set-based formulation. Thus, currently, the methods used for fitness assignment and selection result mostly from the adaptation of the methods used in well-known EMOAs such as SPEA2 and NSGA-II. These adapted methods resemble the two-step selection frequently used to rank solutions in traditional multiobjective optimization before the introduction of quality indicators. This two-step selection consists of a step to select “the best” solutions followed by step to impose diversity, or to break ties, based on crowding. The selection methods used in [45, 7, 6, 9] are briefly described and discussed next.

In the proposal by Carrano *et al.* [45], the ranking of solutions is based firstly on splitting them into two sets, one containing all solutions that map to, at least, one nondominated point (considering all points associated to all solutions) and another containing all remaining solutions. Any solution in the first set is preferred to any other in the second set. The second criterion for sorting solutions in each of the two sets is based on a modified version of dominance strength from SPEA2. In SPEA2 the raw fitness of a solution corresponds to the raw fitness of the single point which the solution is mapped to (see Section 2.4.1 on “Pareto-based methods” for more details). In the modified version, the raw fitness is the sum of the raw fitness of all points in objective space in which the solution is mapped.

Using the method by Carrano *et al.* [45], the solutions in the example in Figure 5.12 are first divided in sets $\{\mathrm{C}, \mathrm{D}\}$ and $\{\mathrm{A}, \mathrm{B}\}$ (the solutions of the first are the preferred ones) and then ranked inside each set by their raw fitness. Thus, the final rank of solutions is: $\mathrm{C}, \mathrm{D}, \mathrm{A}, \mathrm{B}$. Although all solutions mapping into, at least, one nondominated point among all solutions, are preferred to any other that does not, it is possible to see that a dominated solution might be strictly preferred to the solution that dominates it w.r.t. set-dominance. For example, solutions mapping to, at least, one of the following points are the preferred ones: $c^{(1)}, c^{(2)}, c^{(3)}, d^{(1)}$. That is, solutions C and D. However, solution A is preferred (has lower raw fitness) to B although B dominates A.

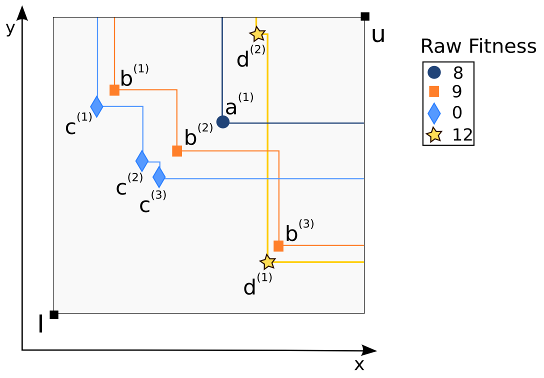


Figure 5.12: Example of the raw fitness as proposed by Carrano *et al.* [45] for four solutions whose corresponding point sets in two-objective space are depicted. The four point sets are $\mathrm{A} = \{a^{(1)}\}$, $\mathrm{B} = \{b^{(1)}, b^{(2)}, b^{(3)}\}$, $\mathrm{C} = \{c^{(1)}, c^{(2)}, c^{(3)}\}$, and $\mathrm{D} = \{d^{(1)}, d^{(2)}\}$ which are represented by circles, squares, diamonds and stars, respectively.

Avigad *et al.* [6, 7] proposed a method based on NSGA-II for comparing/evaluating the quality of solutions for the worst-case robustness problem. The proposed method is also a two-step method to rank solutions. The first step is based on the nondominated sorting of NSGA-II to rank groups of solutions and then an adapted version of the additive ϵ-indicator is used to rank solutions in the same rank group. However, the nondominated sorting is based on a relation between solutions which is not compatible with set-dominance due to the problem's characteristics. Therefore, it will not be further detailed.

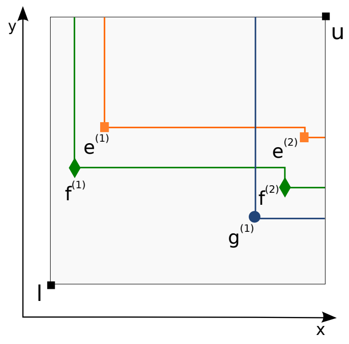


Figure 5.13: Example of the point sets in the two-dimensional objective space associated to three solutions. The three point sets are E $= \{e^{(1)}, e^{(2)}\}$, F $= \{f^{(1)}, f^{(2)}\}$ and G $= \{g^{(1)}\}$ which are represented by squares, diamonds and circles, respectively.

In one of the environmental selection methods used by Bader *et al.* [9], solutions are ranked and then selected based on the hypervolume indicator value associated to the point set to each solution is mapped. In that case, it may happen that a solution mapped to a point not dominated by any other solution is ranked worse than a dominated solution. This can be observed in the example of Figure 5.13, where, based on this method, solutions are ranked in the following order: F, E, G. Thus, if subset selection (environmental selection) is to be performed to select two of them, the clearly dominated solution (E) is preferred over a nondominated solution (G).

Discussion

From a critical point-of-view, the method by Carrano *et al.* [45] for ranking solutions focuses mainly on the individual quality of a point in the objective space rather than on the quality of the set in which it belongs to. As a consequence, the good/bad evaluation of a solution may be highly influenced by a single point in the corresponding set. Moreover, a nondominated point set (e.g., B) may be considered worse than one it dominates (e.g., A). On the other hand, although the quality of the point set is taken into account in a ranking method based on a quality indicator, such as the method described in [9], as a subset selection method it may still contradict the Pareto dominance notion extended to sets of points sets.

It is reasonable to admit that in cases where, at the end, only the best point in the set is selected by the DM, focusing on the individual quality of the points in objective space associated with a solution, as in [45], may be desirable. However, if, in the end, the DM's decision relies on a subset or the complete set of points associated with a solution, then important solutions or even optimal ones can be easily lost. For example, in Figure 5.12, B might be more interesting to a DM interested in the whole set than C or D. Moreover, focusing on point-set quality and not accounting for the quality of the set of solutions (set of point sets) as a whole while performing environmental selection may lead to convergence to sub-optimal solutions due to loss of diversity.

Subset selection, which is relevant to the environmental selection step in EAs as performed in the above methods, i.e., based on ranking, resembles the methods used in EMOAs before the prominence of quality indicators such as the hypervolume indicator. Quality indicators stood out because they implicitly favor good and diverse sets of solutions by considering the quality of the set of solutions as a whole and not the quality of individual solutions by themselves. That has allowed the properties, such as monotonicity and optimal μ-distributions, to be studied and understood. It remains unknown to what extent the selection methods for SVO problems described above possess such properties.

5.6.2 Formulation

Let us consider the set-valued optimization (SVO) problem as the problem of minimizing a function $f : \Omega \to 2^{\mathbb{R}^d}$, where Ω denotes the decision space, and the target space is the power set of the objective space $\mathbb{R}^d$. The SVO problem provides a more general formulation of the traditional multiobjective optimization problem where the image A of a solution is now a non-empty finite point set instead of a single point. Therefore, to accommodate SVO problems, the underlying uncertainty model of the HSR indicator has to be extended. In this case, the DM preferences are also expressed in terms of a single goal vector assumed to follow a uniform distribution over an orthogonal range $[l, u]$ in objective space, where $l, u \in \mathbb{R}^d$. Moreover, individual return (or acceptability) is also a Bernoulli random variable which takes a value of 1 or 0 depending on whether the corresponding individual solution is acceptable or not, respectively. The main differences with respect to the HSR indicator are the definition of image of a solution and of what is an acceptable individual. These definitions are extended as follows:

- There are $n \geq 1$ individuals in the population. To each individual i, $i = 1, ..., n$, corresponds a non-empty and finite set $\mathrm{A}^{(i)}$ of $m^{(i)}$ nondominated objective vectors, i.e., $\mathrm{A}^{(i)} \subset \mathbb{R}^d$, such that $|\mathrm{A}^{(i)}| = m^{(i)} \geq 1$.
- Individuals are either acceptable or not acceptable, depending on whether or not there is a point in the corresponding set that weakly dominates the random goal vector, respectively.

Note that the preference model used in the HSR indicator (see Section 2.4.5) is a special case of the above model where $m^{(i)} = 1$ for all $i \in \{1, ..., n\}$.

The expected return r_i of an individual i as described above is:

$$r_i = p_i = \frac{\Lambda\left([l,u] \cap \left(\bigcup_{a \in \mathrm{A}^{(i)}} [a, \infty[\right)\right)}{\Lambda([l,u])} \tag{5.55}$$

i.e., r_i is the proportion of the objective space in $[l, u]$ which is dominated by $\mathrm{A}^{(i)}$. The probability to simultaneously accepting two individuals i and j is proportional to the measure of the objective space in $[l, u]$ which is simultaneously dominated by both $\mathrm{A}^{(i)}$ and $\mathrm{A}^{(j)}$:

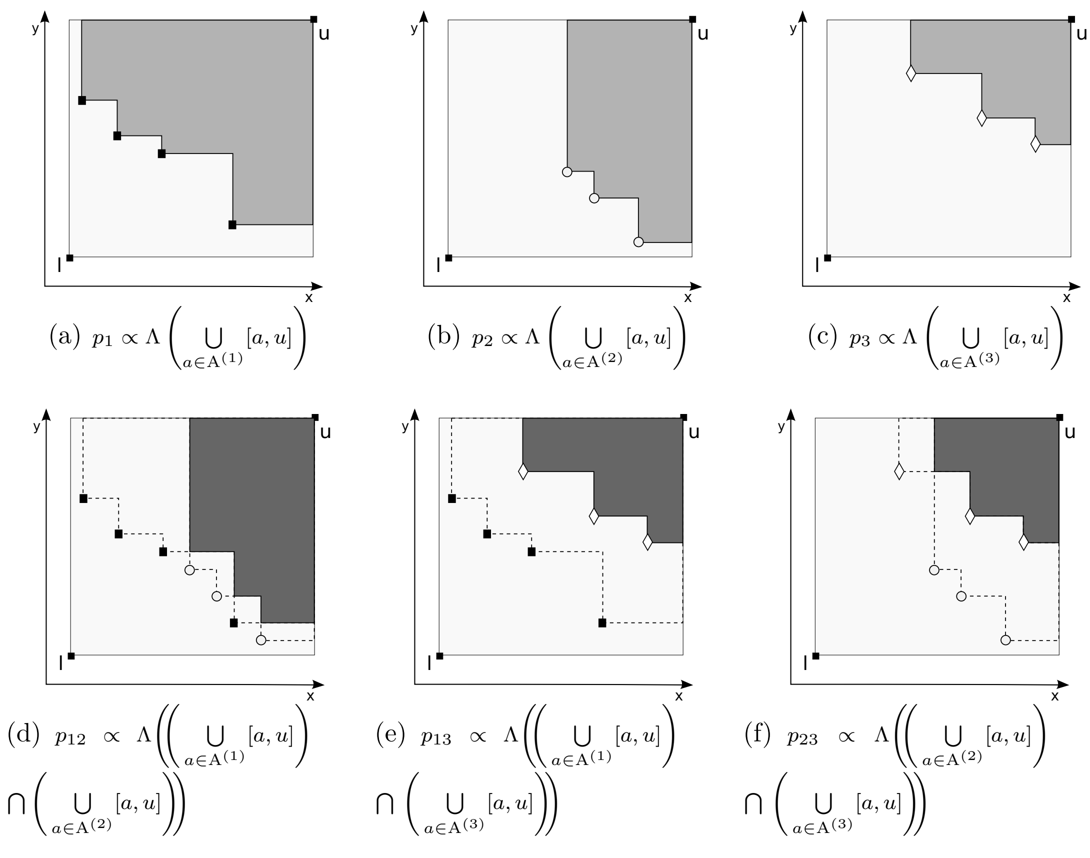
$$p_{ij} = \frac{\Lambda\left([l,u] \cap \left(\bigcup_{a \in \mathrm{A}^{(i)}} [a, \infty[\right) \cap \left(\bigcup_{b \in \mathrm{A}^{(j)}} [b, \infty[\right)\right)}{\Lambda([l,u])} \tag{5.56}$$

As before, the covariance between individuals (i) and (j) is given by:

$$q_{ij} = p_{ij} - p_i p_j \tag{5.57}$$

and $r_f = 0$ under such a model.

Figure 5.14 considers a set of three individuals $\{\mathrm{A}^{(1)}, \mathrm{A}^{(2)}, \mathrm{A}^{(3)}\}$ and shows, in gray, the region measured to compute p_i and p_{ij}, where $i, j \in \{1, 2, 3\}$, assuming, without loss of generality, that $l = (0, 0)$ and $u = (1, 1)$. In the example, $m^{(1)} = 4$, $m^{(2)} = 3$ and $m^{(3)} = 3$.



(a) $p_1 \propto \Lambda\left(\bigcup_{a \in \mathrm{A}^{(1)}} [a, u]\right)$ (b) $p_2 \propto \Lambda\left(\bigcup_{a \in \mathrm{A}^{(2)}} [a, u]\right)$ (c) $p_3 \propto \Lambda\left(\bigcup_{a \in \mathrm{A}^{(3)}} [a, u]\right)$

(d) $p_{12} \propto \Lambda\left(\left(\bigcup_{a \in \mathrm{A}^{(1)}} [a, u]\right) \cap \left(\bigcup_{a \in \mathrm{A}^{(2)}} [a, u]\right)\right)$ (e) $p_{13} \propto \Lambda\left(\left(\bigcup_{a \in \mathrm{A}^{(1)}} [a, u]\right) \cap \left(\bigcup_{a \in \mathrm{A}^{(3)}} [a, u]\right)\right)$ (f) $p_{23} \propto \Lambda\left(\left(\bigcup_{a \in \mathrm{A}^{(2)}} [a, u]\right) \cap \left(\bigcup_{a \in \mathrm{A}^{(3)}} [a, u]\right)\right)$

Figure 5.14: An example of the region measured to compute p_{ij}, given the individuals $\mathrm{A} = \{\mathrm{A}^{(1)}, \mathrm{A}^{(2)}, \mathrm{A}^{(3)}\}$ where $\mathrm{A}^{(i)} \subset \mathbb{R}^2$ and $i = 1, 2, 3$. The points in $\mathrm{A}^{(1)}$, $\mathrm{A}^{(2)}$ and $\mathrm{A}^{(3)}$ are represented as squares, circles and diamonds, respectively. The region measured to compute p_1, p_2 and p_3 is depicted in gray in Figures (a), (b) and (c), respectively, and the region measured to compute p_{12}, p_{13} and p_{23} is depicted in dark gray in Figures (d), (e) and (f), respectively.

Although general, this model may be computationally expensive. Computing the probability p_i requires computing the hypervolume indicator of $m^{(i)}$ points, while computing the probability p_{ij} for $i \neq j$ requires computing first the point set defining the second layer of the Empirical Attainment Function (EAF) [67] (such point set defines the region simultaneously attained by the two sets), and then the hypervolume indicator of such point set. Although computing all of this requires linear time in $m^{(i)} + m^{(j)}$ for $d = 2$ after sorting, for the more general case it requires computing the hypervolume indicator in exponential time in d of a set of $\Omega((m^{(i)} + m^{(j)})^{\lfloor d/2 \rfloor})$ points [67]. Therefore, the applicability of this model may be somewhat limited to a low number of dimensions and/or low number of points associated with each individual/solution. Still, it is worth studying this model as it may provide advantages that other selection methods/indicators do not.

The extended uncertainty model for SVO problems leads to a new instance of the Sharpe-ratio indicator (see Definition 5.1) which is a more general version of the HSR indicator (see Definition 5.2). It is formally defined as:

Definition 5.4 (Extended Hypervolume Sharpe-Ratio Indicator). Let $\mathrm{A} = \{\mathrm{A}^{(1)}, ..., \mathrm{A}^{(n)}\} \subset 2^{\mathbb{R}^d}$ be a non-empty set where $\mathrm{A}^{(i)}$ denotes a finite non-empty nondominated point set, $i \in \{1, ..., n\}$. Given A, the points $l, u \in \mathbb{R}^d$ such that $l \ll u$, and the Sharpe ratio function $h^{\mathrm{A}}_{EHSR}(x, l, u)$ where the expected return p and the covariance Q are computed as expressed in (5.55) to (5.57), the extended hypervolume Sharpe-ratio indicator $I_{EHSR}(\mathrm{A}, l, u)$ is given by:

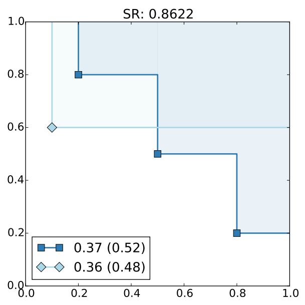
$$I_{EHSR}(\mathrm{A}, l, u) = \max_{x \in \Omega} \ h^{\mathrm{A}}_{EHSR}(x, l, u) \tag{5.58}$$

where $\Omega \subset \mathbb{R}^n$ is the set of solutions that satisfy the constraints of Problem 2.2.

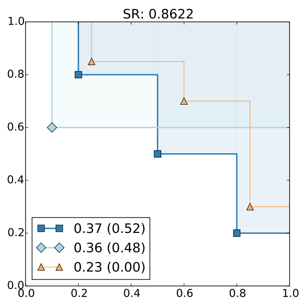
5.6.3 Experiments and Discussion

A lot has to be considered when working with problems where each solution maps to a single point in the objective space: what is a good solution, how are known preferences expressed, if any, what is a good and diverse set of solutions, and so on. Considering that each solution maps to a set of (nondominated) points does not make it any easier to answer these questions, on the contrary. Hence, only some small experiments will be presented in this section. The aim is to give some insight on how investment is assigned by the EHSR indicator, just to have an idea of what to expect and to raise interest in the new indicator. A more thorough experimental and theoretical studies is left for future work.

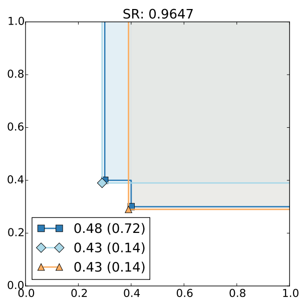
Figure 5.15 shows a few small examples of how investment is distributed by the EHSR indicator. In the first example (Figure 5.15(a)) considering two solutions, a higher investment is assigned to the solution with higher expected return. Adding a dominated solution to this set of solutions does not affect their investment as the dominated solution is assigned zero investment (see Figure 5.15(b)). This observation is consistent with the HSR indicator, for which every dominated solution is assigned zero investment (as proved in Lemma 5.4). Thus, it is reasonable to conjecture that this also holds for the more general EHSR-indicator case. Another scenario is presented in Figure 5.15(c), where no solution is dominated. However, the two points of one solution (marked with squares) are dominated by the union



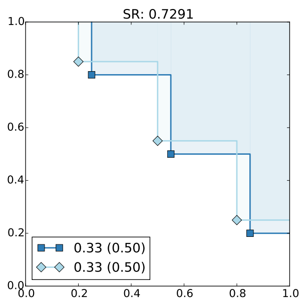
(a)



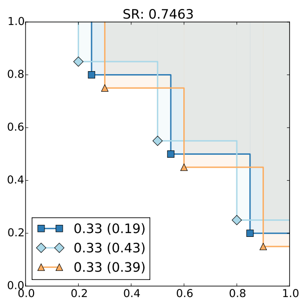
(b)



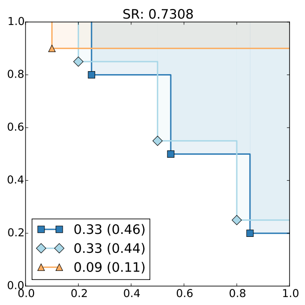
(c)



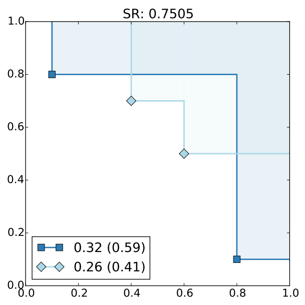
(d)



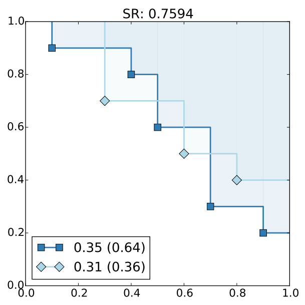
(e)



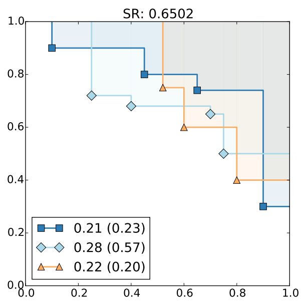
(f)



(g)



(h)



(i)

Figure 5.15: Examples of the EHSR indicator in the two-dimensional case. The figures show how the optimal investment is distributed between the two/three solutions considered. The set of points (in the objective space) corresponding to a solution (in the decision space) share the same color and marker. The region dominated by such set is delimited and shaded in that color. The labels indicate the expected return of each solution/point set (the area dominated) and, in round brackets, the investment assigned to that set. The Sharpe ratio is shown in the title.

of the points of the remaining solutions. Nevertheless, the two-point solution is assigned greater investment (0.72) than the others. Note that the most valued solution in this example is the one that the method by Carrano *et al.* [45], based on raw fitness, would disregard (see Section 5.6.1). The EHSR indicator may be valuing more nondominated solutions that dominate more space in $[l, u]$.

Figures 5.15(d) to 5.15(f) explore cases where there are different solutions with equal expected return. In the first case (Figure 5.15(d)), the optimal investment is split equally between the two solutions. In the second case (Figure 5.15(e)), which adds a third distinct solution to the example of Figure 5.15(d) with equal expected return, the same cannot be observed. In this case, the solution whose exclusive dominated region is the smallest (represented with squares) is assigned lower investment (only 0.19). The remaining investment is unevenly distributed between the other two solutions, where the solution with higher exclusive dominated region has higher investment. If the third nondominated solution added to Figure 5.15(d) had considerably less expected return, it would still receive investment, but in this case (Figure 5.15(f)) less investment would be withdrawn from the other solutions. The solution that loses more investment is the one that has a larger amount of its dominated region become dominated by the third solution. Figures 5.15(g) to 5.15(i) show more examples which seem to be in accordance to the above observations.

The results presented indicate that (some) dominated solutions are assigned zero investment. The assignment of strictly positive investment seems to be related to the size of the region dominated (the more, the better) but it also seems to depend on how much of that region is dominated by other solutions (the less, the better). These observations give a good idea of what to expect from the EHSR indicator and do show that it might provide novelty to the way SVO problems are solved, in particular, in what comes to selecting a diverse set of solutions. It can be used for both environmental selection as well as for fitness assignment, as the HSR indicator, and it seems to implicitly invest in good solutions and in diverse sets of solutions as well. Therefore, it is worth studying the EHSR indicator more thoroughly in the future.

5.7 Preferability Hypervolume Sharpe-Ratio Indicator

In the underlying uncertainty model of the HSR indicator, DM preferences are expressed as a random goal vector in objective space, where a solution is acceptable to the DM if it (weakly) dominates such goal vector. Thus, computing the expected return, i.e., the probability of a solution being acceptable, becomes equivalent to computing the proportion of random goal vectors it dominates (assuming a uniform distribution). Therefore, such a preference model is constructed upon the Pareto-dominance relation. However, such a relation is not adequate for constrained multiobjective optimization problems, as solutions cannot be compared based solely on objective values. For example, it is typically accepted that an infeasible solution should not be preferred over a feasible one, independently of their objective values. Therefore, the preference model of the HSR indicator has to be extended to account for the feasibility of solutions [133]. A way to do that is to extend the model by replacing the underlying dominance relation with another outperformance relation [87]. Such a relation should be Pareto-dominance compliant over feasible solutions, and should reflect preferences also over infeasible solutions. For that purpose, the preferability relation [66] (see Preference Information in Section 2.3.2) will be considered.

The resulting model allows preferences to be expressed in such a way that feasible solutions are preferred to infeasible solutions, and among infeasible solutions, those that are closer to satisfying all (inequality) constraints are preferred. This leads to a new instance of the Sharpe-ratio indicator class that deals with infeasible solutions in an integrated way[1]. The new indicator, that will be called Preferability HSR indicator (PHSR indicator), is identical to HSR indicator but where the computation of the expected return (and covariance) is, in part, different. The Sharpe-ratio indicator can then be directly computed without any further changes.

5.7.1 Formulation

Consider a constrained optimization problem consisting in minimizing d objective functions subject to n^c constraints expressed as $c(x) \leq g$, where $c : \Omega \to \mathbb{R}^{n^c}$ and where Ω, $\mathbb{R}^d$ and $\mathbb{R}^{n^c}$ denote the decision, the objective and the constraint spaces, respectively. The (goal) vector $g = (g_1, \ldots, g_{n^c})$ defines the constraint bounds.

To replace the dominance relation in HSR indicator by the preferability relation, the space defined by the Cartesian product between objective and constraint spaces, $\mathbb{R}^d \times \mathbb{R}^{n^c}$, has to be considered. DM preferences are now expressed in terms of a random goal vector drawn from the uniform distribution on an orthogonal range $[l, u] \times [l^c, u^c] \subset \mathbb{R}^d \times \mathbb{R}^{n^c}$, where $l, u \in \mathbb{R}^d$ and $l^c, u^c \in \mathbb{R}^{n^c}$ such that $l \lll u$ and $l^c \lll u^c$. The *preference vector* for a constrained optimization problem is given by the low priority goal vector $(-\infty, \ldots, -\infty)$ for the objectives and by the high priority goal vector g for the constraints [66]. Therefore, two vectors in $[l, u] \times [l^c, u^c]$ are first compared based on their constraint values, and only if those are satisfactory they are then compared based on their objective values. Therefore, an individual is either *acceptable* or *not acceptable*, depending on whether it is weakly preferred to the random goal vector given the preference vector. As for the HSR indicator, the return of an individual can be seen as a Bernoulli random variable, and the expected return is the probability of being acceptable. In this case, this is equivalent to the proportion of random goal vectors that are worse than the individual considered according to the preferability relation.

Consider a population $\mathrm{A} = \{a^{(1)}, ..., a^{(n)}\} \subset \mathbb{R}^d \times \mathbb{R}^{n^c}$ with n individuals, here represented by their mapping onto objective and constraint spaces. Let $(a, a^c) = (a_1, \ldots, a_d, a^c_1, \ldots, a^c_{n^c}) \in \mathrm{A}$ be one such individual. For simplicity, assume that $a \in [l, u]$ and $a^c, g \in [l^c, u^c]$. Consequently, if (a, a^c) is an *infeasible* solution, let $\mathrm{I}^{\smile}$ and $\mathrm{I}^{\frown}$ be a partition of the set $\{1, \ldots, n^c\}$ denoting the constraint indexes satisfied and not satisfied by a^c, respectively. Then, according to the preferability relation, the expected return, r, of such an infeasible solution (a, a^c) is given by:

$$r = p = \frac{\Lambda(\prod_{i \in \mathrm{I}^{\smile}} [l^c_i, u^c_i] \times \prod_{i \in \mathrm{I}^{\frown}} [a^c_i, u^c_i] \times [l, u])}{\Lambda([l^c, u^c] \times [l, u])} \tag{5.59}$$

where $\times$ denotes the Cartesian product. If (a, a^c) is a *feasible* solution instead, then the expected return of (a, a^c), r, is given by:

$$r = p = \frac{\Lambda(([l^c, u^c] \backslash [l^c, g]) \times [l, u]) + \Lambda([l^c, g]) \times [a, u])}{\Lambda([l^c, u^c] \times [l, u])} \tag{5.60}$$

The probability to simultaneously accepting two solutions is computed in an analogous way, where a and a^c are replaced with the component-wise maximum between

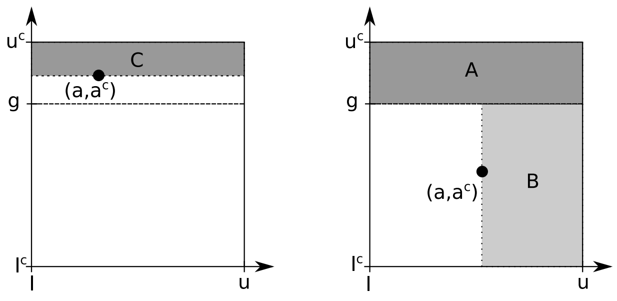
[1]This model was used in the experiments by Yevseyeva et al. [133] but without detailing it. This section formalizes and details how the expected return is computed.

the objective vectors and between the constraint vectors of the two solutions, respectively. With the probability of accepting an individual, and of simultaneously accepting two individuals, the covariance matrix can then be computed as for the HSR indicator and the EHSR indicator (see expression(5.57)). Given that $r_f = 0$ under such a model, and the above definition, the new indicator can now be formalized:

Definition 5.5 (Preferability Hypervolume Sharpe-Ratio Indicator). Given a non-empty set $\mathrm{A} = \{\mathrm{a}^{(1)}, ..., \mathrm{a}^{(n)}\} \subset \mathbb{R}^d \times \mathbb{R}^{n^c}$, the points $l, u \in \mathbb{R}^d$ where $l \ll u$, the points $l^c, u^c, g \in \mathbb{R}^{n^c}$ where $l^c \ll u^c$, and the Sharpe ratio function $h^{\mathrm{A}}_{PHSR}(x, l, u, l^c, u^c, g)$ where the expected return and the covariance matrix are computed as expressed in (5.59), (5.60) and (5.57), the preferability hypervolume Sharpe-ratio indicator $I_{PHSR}(\mathrm{A}, l, u, l^c, u^c, g)$ is given by:

$$I_{PHSR}(\mathrm{A}, l, u, l^c, u^c, g) = \max_{x \in \Omega} \; h^{\mathrm{A}}_{PHSR}(x, l, u, l^c, u^c, g) \tag{5.61}$$

where $\Omega \subset \mathbb{R}^n$ is the set of solutions that satisfy the constraints of Problem 2.2.



(a) Infeasible solution (b) Feasible solution

Figure 5.16: An example of the region measured, in $[l, u] \times [l^c, u^c] \subset \mathbb{R}^d \times \mathbb{R}^{n^c}$ space, to compute the expected return of a feasible and an infeasible solution with objective value a and constraint value a^c. The objective and the constraint correspond to the x and y-axis, respectively, where g denotes the constraint goal.

Figure 5.16 shows a simple example of the area measured when computing the expected return of an infeasible solution (Figure 5.16(a)) and of a feasible solution (Figure 5.16(b)), considering a single objective ($d = 1$) and a single constraint ($n^c = 1$). Assume, without loss of generality, that $\Lambda([l^c, u^c]) = \Lambda([l, u]) = 1$. The expected return of an infeasible solution (a, a^c) is the measure of the region containing all infeasible solutions (i.e., all goal vectors) that are worse than (a, a^c), in this case represented by region C. That is, a solution is worse than (a, a^c) if it violates all constraints that (a, a^c) violates (i.e., constraints $\mathrm{I}^{\frown}$) at least as much, but not equally in all cases, independently of how it performs in the constraints that (a, a^c) satisfies (i.e., constraints $\mathrm{I}^{\smile}$) and in the objective space. Although a solution that equally violates all constraints that (a, a^c) violates may also be worse than (a, a^c) according to the preferability relation, the set of such solutions has zero measure in constraint space, and does not affect the probability of (a, a^c) being acceptable to the random DM. The expected return of a feasible solution (a, a^c) is the measure of the region containing all infeasible solutions (region A) plus the region containing all feasible solutions that a dominates (region B).

5.8 Discussion and Concluding Remarks

The Sharpe-ratio indicator class has been formalized, and theoretical results on the particular HSR indicator have been presented regarding independence of one of the reference points, scaling independence, monotonicity properties, and optimal μ-distributions on two-objective linear fronts.

Although the formulation of the HSR indicator involves two reference points, only one needs to be set in practice. The second reference point is just a technical parameter that is required by the formulation. Indeed, the optimal investment is not affected by the linear objective rescaling implied by changes to this second reference point, and the indicator is scaling independent under such transformations. Thus, the HSR indicator does not require more parameters to be set than, for example, the hypervolume indicator.

Furthermore, the HSR indicator is both weakly monotonic with respect to weak set dominance and strictly monotonic with respect to elementwise set dominance, the latter property guaranteeing that at least one nondominated point is included in any indicator-optimal subset. More specifically, the HSR indicator is such that dominated points are always assigned zero investment, and at least two nondominated points are assigned strictly positive investment if they exist in a set. As a consequence, the population of an indicator-based EMOA implementing environmental selection based on the HSR indicator will in fact tend to move closer to the Pareto front as new nondominated points are found.

Finally, the optimal μ-distribution of the HSR indicator on two-objective linear fronts is unique and identical to that of the hypervolume indicator, where all points are equally spaced between two outer points, and the location of the outer points depends on the position of the front relatively to the (upper) reference point. In contrast to the hypervolume indicator, however, the HSR indicator assigns to each point in a set a weight (investment) that can be interpreted as fitness and used to perform parental selection in EAs. Still for a two-objective linear front, the optimal investment in the inner points turns out to be proportional to the distance between their two direct neighbours regardless of where in between those neighbours each point lies, which resembles the concept of crowding distance in NSGA II [53]. Since all inner points in the optimal μ-distribution are equally crowded, the corresponding investment/fitness is the same for all of them. On the other hand, the outer points are assigned greater or equal investment than the inner points. The farther away from the front the upper reference point is, the greater the investment in the outer points of the optimal μ-distribution, as if to promote the expansion of the front beyond its boundaries. This suggests that it may be desirable to adjust the upper reference point during the execution of an HSR indicator-based EMOA.

The experimental results on the HSR indicator extend the understanding of this indicator to other than linear fronts. The results clearly show a bias towards knee regions and extreme points, which is accentuated the farther away the upper reference point is. This is merely a reflection of the underlying preference model, where the effect of setting the upper reference point far away can be viewed as a way to explicitly give less importance to a solution which improves in an objective less than it loses in the other in comparison to its neighbors. From a practical point-of-view in EMO, the characterization of the reflected preferences over sets of the HSR indicator is a very important result taken from this chapter as it allows to better understand the indicator and make a better use of it.

Besides providing tools for the theoretical and experimental characterization of Sharpe ratio based indicators, this chapter also provides examples of two other instances of this class of indicators. These instances show how problems with different characteristics can be considered in an integrated way. These indicators explore just part of the potential and the flexibility of the class of Sharpe-ratio indicators for preference integration. For instance, the different ways of expressing preferences studied in MCDA could be considered.

Overall, this chapter opens up several paths for future work. Firstly, it shows how the HSR indicator can be theoretically and experimentally characterized. Such tools can be used to extend the results on HSR indicator for the 3-dimensional case and/or for other instances of the Sharpe-ratio indicator class, such as the PHSR indicator. Note that, proving a variant of Lemma 5.4 where the dominance relation is replaced by another Pareto-compliant relation, such as the preferability relation (PHSR indicator), should suffice to show that a Sharpe-ratio based indicator based on such a relation has the same monotonic properties as the HSR indicator. A worthy future study is on whether the EHSR indicator propagates the monotonic properties of the HSR indicator to sets of solutions where each solution is represented by a nondominated point set. Finally, the validation of the HSR indicator opens up a path for the formulation of other instances for both cases of known and unknown preference information.

Chapter 6

Portfolio Selection in EMOAs: Experimental Results

An immediate consequence of the algorithms to compute/update contributions proposed in Chapter 4 is the runtime improvement of $(\mu + 1)$ EMOAs based on the exact solution of the HSSP, such as SMS-EMOA, in three and four-objective cases. However, using such exact algorithms in the general $(\mu + \lambda)$ case is still too expensive. The greedy incremental/decremental approaches to the HSSP were shown to produce good approximations, and in a short amount of time with the algorithms proposed in Chapter 4. This indicates that they can be a good alternative to exact algorithms, which can make hypervolume-based environmental selection more affordable and appealing, at least in $d = 3, 4$. Moreover, the theoretical properties of the HSR Indicator shown in Chapter 5, its computational cost and its formulation make it an interesting indicator for EMOAs. As discussed in [133], the portfolio formulation naturally suggests that the HSR indicator can be used, in an integrated way, for both environmental selection and fitness assignment. Consequently, the next natural step is to integrate the proposed greedy algorithms for the HSSP and the HSR indicator in EMOAs, and study their performance on benchmark problems.

The intent of this chapter is not to show which EMOA, with which operator combination, is the best. The goal is merely to study the suitability for environmental selection in $(\mu + \lambda)$-EMOAs of Archiving Algorithms (AAs) based either on the hypervolume indicator or on the HSR indicator, and to study how their behavior differs for different settings of λ in problems with up to four dimensions. Moreover, in the case of hypervolume-based AAs, the goal is mainly to study the impact of using a (either incremental or decremental) greedy strategy to approximate the HSSP instead of solving it exactly, and whether the results observed in Section 4.6 for predetermined sequences of (nondominated) solutions are also observed in a more stochastic environment provided by the inclusion of variation operators. Therefore, the EMOAs and the experimental setup were constructed so as to keep the experiments as controlled as possible, with the aim of having results more dependent on the environmental selection method used, and less on stochastic aspects.

Four EMOAs using different environmental selection and/or fitness assignment methods were implemented and evaluated. The EMOAs will be described next in Section 6.1, and all the implementation details specific to the optimization strategy (e.g., how invalid and dominated solutions were dealt with) are detailed as well. The benchmark problems used for testing the EMOAs are then described in Section 6.2, as well as the implementation details specific to those problems (e.g., representation, variation operators). Finally, the results are presented and discussed in Section 6.3 and concluding remarks are drawn in Section 6.4.

6.1 EMO Algorithms

Four $(\mu + \lambda)$-EMOAs were tested, namely:

- HSSP-EMOA (only for d=2)
- gHSS-EMOA (only for d=2,3)
- gHSSD-EMOA
- POSEA

The first three algorithms are alike and only differ in the (nondecreasing) archiving algorithm used for environmental selection. HSSP-EMOA uses an exact algorithm for the HSSP for environmental selection, while the remaining two use, as the names suggest, the two greedy strategies to approximate the HSSP discussed in Chapter 4, the incremental (gHSS) and decremental (gHSSD), respectively. The particular cases of HSSP-EMOA and of gHSSD-EMOA with $\lambda = 1$ are equivalent to SMS-EMOA, and thus will work as a baseline. POSEA (Portfolio Optimization Selection Evolutionary Algorithm) uses the HSR indicator discussed in Chapter 5 for environmental selection and fitness assignment. Apart from this and parental selection, POSEA has the same characteristics as the hypervolume-based EMOAs (variation operators, etc).

6.1.1 Implementation Details

Several aspects of the construction and implementation of an EMOA are independent of the problem to be optimized. Some regard the evolutionary algorithm setup:

- the environmental selection method
- the fitness assignment method
- the parental selection method

while others are mainly related to issues related to solution handling:

- how to deal with infeasible solutions
- how to deal with dominated solutions
- how to deal with repeated points (in objective space)

Regarding *environmental selection*, the results observed in Section 4.6 point to a superior performance of *nondecreasing* archiving algorithms compared to their possibly decreasing counterpart. Consequently, the archiving algorithms used for environmental selection are nondecreasing, with respect to both the hypervolume indicator (HSSP-, gHSSD- and gHSS-EMOAs), and the HSR-indicator (POSEA). This means that whenever the selected candidate population has a lower value of the indicator being maximized than that of the last population, then the last population survives instead of the the new candidate one. The implemented EMOAs and the choices made concerning the other aspects are described next.

HSSP- gHSSD- and gHSS-EMOAs

Apart from the algorithm used for *environmental selection*, HSSP- gHSSD- and gHSS-EMOAs share the same operators and implementation details which were chosen based on SMS-EMOA. In the *parental selection*, the λ individuals are randomly selected, where each individual is selected at most $\lceil \lambda/\mu \rceil$ times, which means that the same individual is not selected more than once if $\lambda \leq \mu$. Thus, there is no need for a *fitness assignment* method. In this way, these methods are consistent with the original SMS-EMOA and EMOA performance becomes more dependent on the environmental selection and less on the parental selection and fitness assignment methods, thus resulting in a fairer comparison of environmental selection methods.

HSSP, gHSSD and gHSS algorithms are used in the corresponding EMOAs only when there are more than μ nondominated (feasible) solutions, in which case all *infeasible* and all *dominated* solutions are discarded, and the respective subset selection algorithm is applied to the remaining nondominated (feasible) solutions. Otherwise, *environmental selection* consists in deciding which dominated/infeasible solutions to discard and which to keep. In that case, the oldest (with respect to the generation in which they were added to the population) λ solutions among the infeasible and dominated ones are discarded. Although different from the original SMS-EMOA, this method of discarding dominated/infeasible solutions was used because, as observed in [89], the achieved results are similar and it is computationally less expensive as HSSP/gHSSD/gHSS has to be computed fewer times.

When more than one solution among the $\mu + \lambda$ solutions map to the same point in the objective space (there are *repeated points*), the oldest one is seen as being nondominated while the remaining copies are considered dominated solutions.

POSEA

The HSR indicator is used in POSEA both for *environmental selection* and for *fitness assignment*. The investment assigned by the HSR indicator is directly used as the solution *fitness* in the *parental selection* method, in this case, Stochastic Universal Sampling (SUS) [11]. Ideally, a cardinality constraint would be added to the portfolio selection formulation to forbid more than μ solutions to be assigned strictly positive investment. Although it would be possible to do this, the problem would not be a quadratic programming problem anymore but a mixed integer programming problem instead. To avoid the additional computational effort, the following (relaxed) method was used. After computing the HSR indicator for the $\mu + \lambda$ solutions, solutions are sorted in descending order of the assigned investment, and those assigned equal investment are sorted in ascending order of solution age. The first μ solutions are the selected ones (*environmental selection*). When more than μ solutions are assigned strictly positive investment, the new population may happen to be worse than the previous one, with respect to the HSR indicator. To prevent this, when the highest investment assigned to a discarded solution is higher than a threshold (10^{-6}), the HSR indicator is recomputed for the μ solutions selected. Then, the new population is only accepted if its HSR indicator value does not degrade in comparison to the previous population (working as a nondecreasing archiving algorithm). Recomputing the HSR indicator in such cases also ensures that the investment used later in parental selection as the solution fitness is the optimal investment.

Regarding *repeated points* in objective space, only one occurrence is used in the computation of the HSR indicator (to ensure that the covariance matrix Q has full rank and that the QP problem has a unique solution) and then the investment is split evenly among all copies. The HSR indicator is computed considering the (feasible) nondominated and the *dominated* solutions.

In the case of constrained optimization problems, the population may contain *infeasible* solutions as well. To account for feasible and infeasible solutions under the same model, the preferability relation was used instead of the dominance relation for computing the expected return and return covariance matrix (see Section 5.7).

In addition to POSEA, which was implemented as described above, a variant of POSEA where a limit is imposed on the maximum investment a solution can get and where repeated solutions are treated differently was considered. This variant is identified here as **POSEA-L**, and differs from POSEA in the following. Regarding *repeated solutions* in objective space, only one occurrence is used in the computation of the HSR indicator while the copies are assigned zero investment. Moreover, to ensure that, at least, $\min(\lambda, \mu)$ solutions are assigned positive investment, the investment was limited to a maximum of $1/\min(\lambda, \mu)$ by adding linear constraints to the QP formulation of the HSR indicator. In this way, an individual is selected in the parental selection at most $\lceil \lambda/\mu \rceil$ times. Note that dominated and/or infeasible solutions may be assigned non-zero investment due to this imposition. This is obviously true for a data set containing a single (feasible) nondominated solution and $\lambda > 1$.

6.2 Problem Description

Two types of problem were used in the experiments, continuous and combinatorial optimization problems. The continuous problems used were selected from the DTLZ test suite [54] and the combinatorial problem used was the Multiobjective Knapsack Problem (MKP). These problems will be explained in Sections 6.2.1 and 6.2.2, respectively. Together with the explanation, the implementation details specific to each problem are described, namely, the representation, the generation of the initial population, and the mutation operator used. Note that, to avoid adding more stochastic behaviour, crossover was not used.

6.2.1 DTLZ Fronts

The DTLZ test suite [54] provides a set of seven continuous problems (DTLZ1-DTLZ7) to be used for benchmarking EMOAs in any number of dimensions d. Each problem consists of $m = d + t - 1$ decision variables in $[0, 1]^m$, d objective functions and a function $h : \mathbb{R}^m \to \mathbb{R}$ that, together with the last t decision variables ($x_d, \ldots, x_m$ given a solution $x \in [0, 1]^m$), augments the search space and is used, for example, to introduce many local minima. The objective functions model the Pareto front when h evaluates to zero. The optimal solutions correspond to finding the values of the variables x_d to x_m such that h is minimized ($h(x) = 0$) and the values of the variables x_1 to x_{d-1} that minimize the objective functions. The recommended number of additional decision variables, t, depends on the problem. In these experiments, decision variables were only allowed to take values between

0 and 1 and therefore, all solutions generated by the EMOAs are feasible solutions, which avoids the need to deal with constraints.

Implementation details

Representation Each solution is represented by a real vector in $[0, 1]^m$.

Initial Population All individuals in the initial population are randomly generated by drawing a value between 0 and 1 uniformly at random for each decision variable.

Mutation The mutation operator used was the variable-wise Polynomial Mutation [92]. The parameter η_m required by this operator was set to a standard value of 20. Each decision variable was mutated with a probability of $\frac{1}{m}$.

Objective Space All objective vectors are contained in the orthogonal range $[l, u]$ of the objective space defined by the origin and the coordinate-wise maximum of each objective, i.e., $l = (0, \dots, 0)$ and $u = (M_1, \dots, M_d)$ where M_i for $i = 1, \dots, d$ is the maximum value of each of the objective functions.

Instances To discuss in detail the performance of the EMOAs under consideration, particularly regarding the final populations, only the results for the DTLZ1 and DTLZ2 fronts are presented. The Pareto front of DTLZ1 is a linear front defined by the hyperplane $\sum_{i=1}^{d} f_i(x) = 0.5$ while that of DTLZ2 is a spherical concave front with radius 1 and center at $(0, \dots, 0)$. The parameter t was set to the recommended values, i.e., to $t = 5$ and $t = 10$ for the DTLZ1 and DTLZ2 test problems, respectively. Additionally, the test problem DTLZ2convex is also presented which is analogous to DTLZ2 but has a spherical convex Pareto front instead, where the center of the sphere is at $(1, \dots, 1)$. The following table indicates the maximum function values that define $u \in \mathbb{R}^d$, the upper bound of the orthogonal range in objective space containing all feasible solutions.

problem	u
DTLZ1	(1125,...,1125)
DTLZ2	(3.5, ..., 3.5)
DTLZ2convex	(3.5,..., 3.5)

6.2.2 Multiobjective Knapsack Problem

The Knapsack Problem (KP) is the problem of selecting a set of items to fill a knapsack with limited weight capacity, such that the total item value is maximized. The multiobjective version of this problem [145] considers the simultaneous maximization of the total value of the selected items in each one of d knapsacks, which may all have different capacities. Each item may add a different weight and/or value to each knapsack. The multiobjective knapsack problem is formally defined as follows.

Problem 6.1 (Multiobjective Knapsack Problem). Let m be the number of available items and d the number of knapsacks. Let p_{ij} and w_{ij} denote the (positive) profit and the weight, respectively, of item i in knapsack j, and let W_j denote the maximum capacity of knapsack j, where $i \in \{1, \dots, m\}$ and $j \in \{1, \dots, d\}$. The element x_i of a solution $x \in \{0,1\}^m$ is a binary variable representing whether the item i is selected ($x_i = 1$) or not ($x_i = 0$). Then, solve:

$$\max_{x \in \{0,1\}^m} \quad f_j(x) = \sum_{i=1}^{m} p_{ij} x_i, \qquad j = 1, \dots, d \tag{6.1a}$$

$$\text{subject to} \quad f_j^c(x) = \sum_{i=1}^{m} w_{ij} x_i \leq W_j, \qquad j = 1, \dots, d \tag{6.1b}$$

The objective vector associated with a solution $x \in \{0,1\}^m$ is given by $(f_1(x), \dots, f_d(x))$ while the constraint vector is given by $(f_1^c(x), \dots, f_d^c(x))$ where $f_j(x)$ and $f_j^c(x)$ represents the total value and the total weight of the solution x on knapsack $j \in \{1, \dots, d\}$, respectively. The vector with the capacities of the knapsacks, $g = (W_1, \dots, W_d)$, represents the goal vector g that a solution must attain in order to be considered a feasible solution.

Implementation details

Representation Each solution is represented by a m-sized binary vector.

Initial Population All individuals in the initial population are randomly generated by choosing 0 or 1 uniformly at random for each index.

Mutation The mutation operator used consists of sampling uniformly at random from the 1-flip-exchange neighborhood [104]. A mutation is either a bit flip move (a 0 becomes 1 or vice-versa, i.e., either an item is discarded or an item is selected) or a bit exchange move (a 0 becomes 1 and a 1 becomes 0, i.e., a selected item is discarded and another one becomes selected). Each type of move is applied with a given probability that depends on the number of neighbors (i.e., total number of individuals reachable through a mutation):

$$|N(x)| = m + H(x)[m - H(x)], \qquad \text{where} \qquad H(x) = \sum_{i=1}^{m} x_i$$

where $N(x)$ denotes the set of neighbors of x. A bit flip is performed with probability $\frac{m}{|N(x)|}$, otherwise a bit exchange is performed. The index among the ones/zeros of which bit to flip/exchange is selected uniformly at random.

Objective Space Because the EMOAs were implemented considering minimization problems, the objective values provided to the EMOAs correspond to the negative of the objective functions given each solution. All objective vectors are contained in the orthogonal range $[l, u]$ in objective space defined by minus the coordinate-wise maximum of each objective and the origin, i.e., $l = (-M_1, \dots, -M_d)$ and $u = (0, \dots, 0)$ where $M_j = \sum_{i=1}^{m} p_{ij}$ and $j \in \{1, \dots, d\}$.

Constraint Space All constraint vectors are contained in the orthogonal range $[l^c, u^c]$ of the constraint space defined by the origin and the coordinate-wise maximum of each constraint, i.e., $l^c = (0, \ldots, 0)$ and $u^c = (M_1^c, \ldots, M_d^c)$ where $M_j^c = \sum_{i=1}^{m} w_{ij}$ and $j \in \{1, \ldots, d\}$.

Instances The instances used were taken from an online Test Problem Suite[1]. Here, only the results for the 500-item instances in $d = 2, 3, 4$ are shown. The following table indicates the negative of the total value (in the left column) and the total weight (in the right column) of all items in each knapsack which are used to define the $l \in \mathbb{R}^d$ vector and the $u^c \in \mathbb{R}^d$ vector, respectively. Vectors l and u^c are then used to delimit the considered orthogonal ranges in objective space and constraint space, respectively, for each instance:

d	l	u^c
2	(-27088, -28014)	(27487, 26714)
3	(-26680, -28086, -28090)	(26640, 26260, 27913)
4	(-27069, -27302, -27357, -26957)	(26998, 27510, 26908, 27800)

6.3 Experimental Results

6.3.1 Parameter Settings

The EMOAs were tested for a fixed *population size* of $\mu = 100$. The *number of offspring* considered was $\lambda = 1, \mu/4, \mu/2, \mu, 2\mu$, i.e., $\lambda = 1, 25, 50, 100, 200$. A total of 30 *runs* were considered for each combination of EMOAs, λ value and problem instance. A maximum of 10^6 function evaluations was the *stopping criterion* used for all combination of algorithms and λ value, on every problem considered. The *mutation operator* is applied to every individual selected in parental selection.

The experiments were set in a way that the *initial population* of size μ was the same for all EMOAs for each problem instance, by fixing the random seed for each run, i.e., the same seed was used for the i^{th} run for all algorithms. Consequently, the hypervolume-based EMOAs with the same λ, and on the same problem, will have the same behavior (will select the same population) from the beginning of the run until a generation is reached where environmental selection has to discard nondominated solutions. That is, when the number of nondominated solutions among the population and the offspring is higher than μ. When such a generation is reached, even if two different EMOAs select the same individuals, the following generation is not guaranteed to be equal unless the algorithms used for environmental selection were the same. The reason is that the order of selected individuals provided by the environmental selection may differ and that order will influence the following stochastic steps.

The *reference points* are fixed and set as the minimum/maximum possible objective/constraint values for each problem instance. This ensures that the indicators are always computed with respect to proper reference points, that the indicator values are comparable and that the order imposed by indicators on sets does not change

[1]http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/

throughout the runs execution. On the other hand, the upper reference point may be set too far from the Pareto front which will possibly lead to final populations less uniformly distributed in objective space, but at least they will be distributed in a predictable way justified through the known theoretical results. Recall from the problem descriptions that all objective vectors are contained in the orthogonal range $[l, u]$ where $l, u \in \mathbb{R}^d$ and, in case of constrained problems, all constraint vectors are contained in the orthogonal range $[l^c, u^c]$ where $l^c, u^c \in \mathbb{R}^{n^c}$. Taking into account the considered settings of l, u, l^c and u^c for each instance as described in Section 6.2, for the hypervolume indicator the reference point, r, was set to u, while for the HSR indicator, l and u were used and so were l^c and u^c for the multiobjective knapsack problem.

6.3.2 Results

For every pair of EMOA and λ setting tested, data was collected 200 times throughout each run. The data is always collected after generating the initial population, and once the run is over, while the remaining cases occurred after a fixed number of generations so as to collect the data at even function evaluations intervals. The data collected include the hypervolume and the HSR indicators of the current population, and the number of (feasible) nondominated solutions it contains. For the hypervolume-based EMOAs, the average number of nondominated solutions among the population and offspring in the interval of generations between data point collections is also recorded. For example, if data is collected after the 4^{th} and 7^{th} generation and the number of nondominated solutions among population and offspring in generations 5, 6 and 7 is 110, 130 and 124, respectively, the average in the interval between the 4^{th} and 7^{th} generation collected after the 7^{th} generation is 121.3.

Plots

Most of the results presented here refer to the final populations obtained by the EMOAs. The first set of figures summarizes the information from all 30 runs for a given problem instance. In particular, Figures 6.1 to 6.3 refer to the DTLZ1, DTLZ2 and DTLZ2convex problems in $d = 2$, respectively. Figure 6.17 is analogous but for the multiobjective knapsack problem in $d = 3$. In these figures, each column of plots shows information regarding a particular EMOA, the name of which is indicated in the top plot. The plots in the first two rows of each figure show, as boxplots, the hypervolume indicator (first row), and the HSR indicator (second row), obtained for the 30 final populations. Each plot contains 5 boxplots that refer to the same EMOA but for a different λ setting, specified in the x-axis. The y-axis shows the indicator value. As the axis range is fixed for all plots in the same row, they can all be directly compared. When the indicator values obtained by different EMOAs are too far away, the y-axis is split in two to omit a range of indicator values that were not obtained by any EMOA, and the scale of the two new ranges is kept the same to facilitate the boxplot comparison (e.g., see Figure 6.2).

The bottom row of Figures 6.1 to 6.3 and Figure 6.17, shows the average number of nondominated solutions along runs. For POSEA (leftmost column), these results refer to the number of nondominated solutions in the population at the generation at which the data was collected (averaged across all runs). For the hypervolume-based EMOAs, the results refer to the average number of nondominated solutions among population and offspring in the measured interval of generations previous to the generation at which the data was collected (averaged across all runs).

For every problem instance there is a set of figures showing the final population of the first run of each EMOA for the $d = 2$ or the $d = 3$ case. See Figures 6.4, 6.6 and 6.7 for the $d = 2$ case and Figures 6.8, 6.11, 6.13 and 6.18 for the $d = 3$ case, where the problem instance is identified in the figure caption. Additionally, the 50% Empirical Attainment Function (EAF) [67] for $d = 2$ is presented in Figure 6.5 and, for $d = 3$ in Figures 6.9, 6.12, 6.14 to 6.10 and 6.19. The EAF captures the stochastic behaviour of EMOAs by allowing to study the distribution of an EMOA's outcome in objective space, making it possible to infer the probability that an EMOA has of finding a better solution than any given solution in objective space. The 50% EAF shows the region of the objective space attained by, at least, half of the 30 final populations, i.e., for every point (weakly) dominated by the 50% EAF there were, at least, 15 final populations that dominated it. Pairwise comparisons of the final populations or EAFs are shown only when they can provide additional information. For example, in most of the $d = 2$ cases such plots are not shown because the final populations of EMOAs are overlapped. For $d = 3$, such plots are shown only for the convex shaped fronts because for other fronts, there were overlapping surfaces which were not properly rendered. The final population and EAF plots are shown only for a single setting of λ for each EMOA, in particular, they are shown for POSEA with $\lambda = 100$ and hypervolume-based EMOAs with $\lambda = 1$. The value of λ was selected based on the EMOA overall performance regarding the indicator it tries to maximize, among all problem in all dimensions. Recall that, gHSSD-EMOA with $\lambda = 1$ takes the role of the base case as it is equivalent to SMS-EMOA.

An additional type of plots, let us call it dominance table (see Figures 6.16 and 6.20), is presented. It shows, in a given cell, the average percentage of non-dominated solutions in the final population of an EMOA (in that column cell) that are strictly dominated by the final population of another EMOA (in that row cell). These are computed through pairwise comparison of every EMOA setting for each run. For example, given a problem instance, to compare the final populations obtained by POSEA with $\lambda = 100$ and by gHSS-EMOA with $\lambda = 1$ in their first run, let us assume that POSEA obtained 80 nondominated solution of which gHSS-EMOA dominates 8, while gHSS-EMOA obtained 100 nondominated solutions of which POSEA dominates 25. Then, gHSS-EMOA dominates 10% of POSEA nondominated solutions while POSEA dominates 25% of gHSS-EMOA nondominated solutions. This is computed for the remaining 29 runs and the values are averaged. Only the final populations from the same run index are compared because the EMOAs start with the same initial population, which indicates how they compare to one another when the starting condition is the same.

In this section, only a few representative results are shown. In particular, the boxplots results are shown for all continuous problems in $d = 2$, the final populations are shown for $d = 2$ and $d = 3$ and EAFs are shown only for $d = 3$. For the multiobjective knapsack problem, only a few results are show for $d = 3$. The remaining results, and in particular on $d = 4$, are not shown as they do not provide additional information.

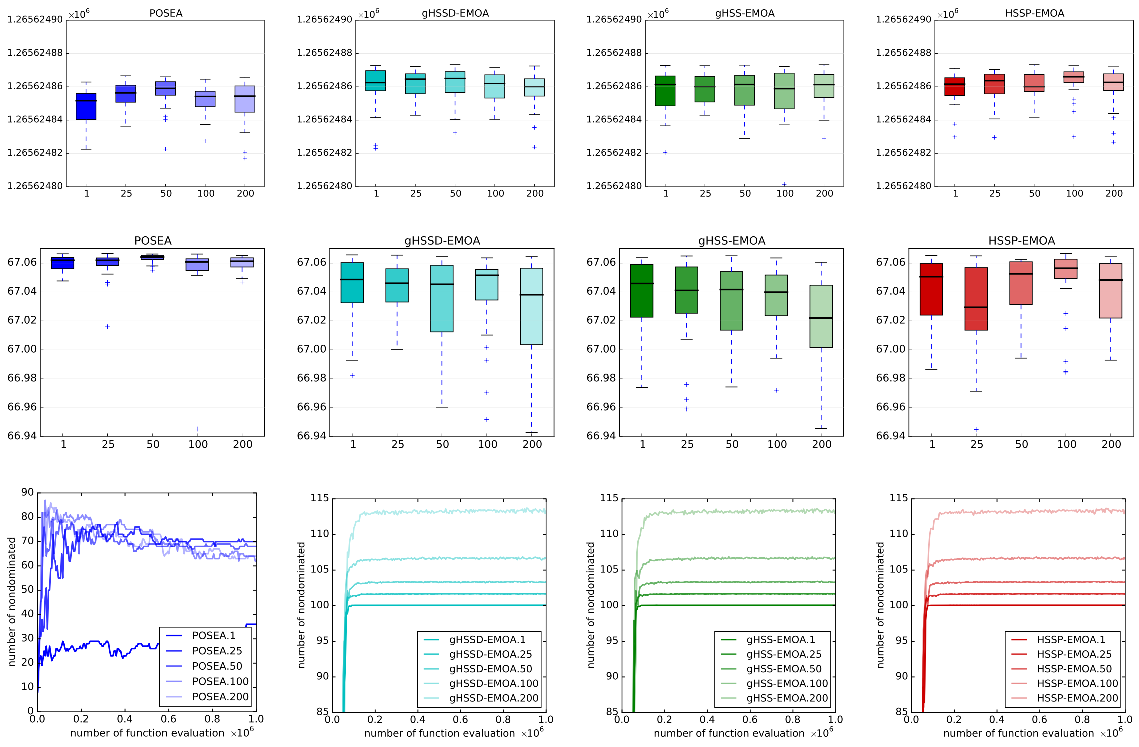


Figure 6.1: Summarized results from 30 runs of POSEA and of gHSSD-, gHSS- and HSSP-EMOAs for DTLZ1 in $d = 2$ (from left to right column, respectively).

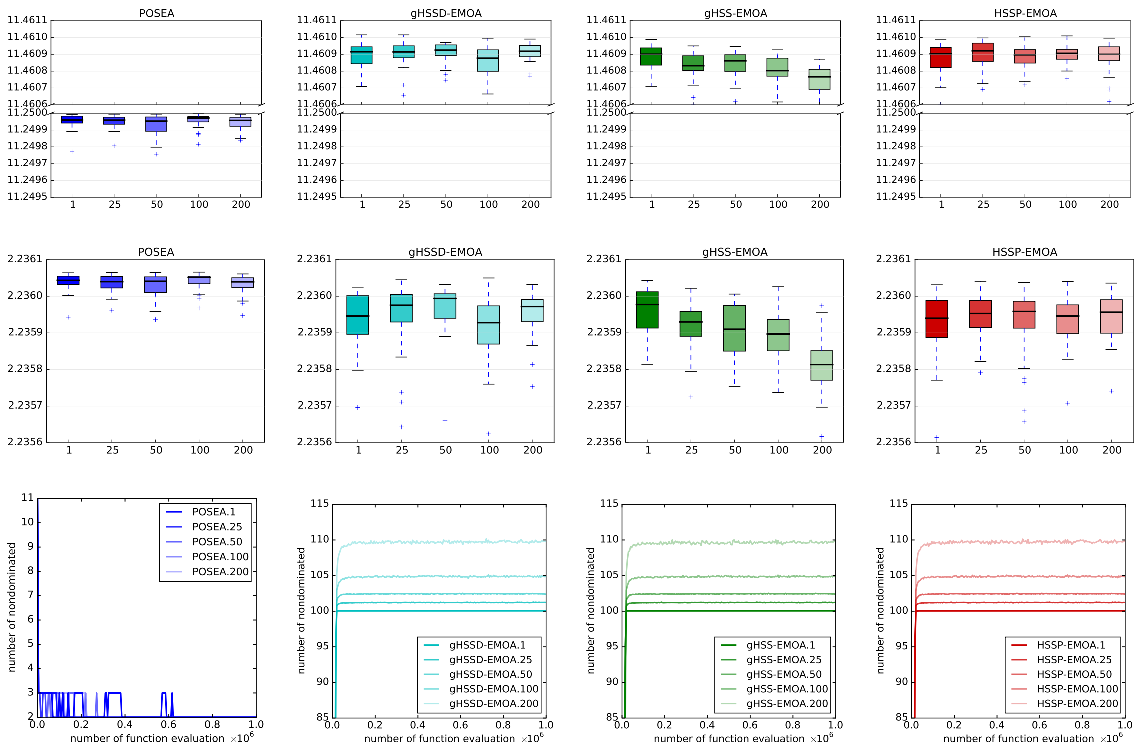


Figure 6.2: Summarized results from 30 runs of POSEA and of gHSSD-, gHSS- and HSSP-EMOAs for DTLZ2 in $d = 2$ (from left to right column, respectively).

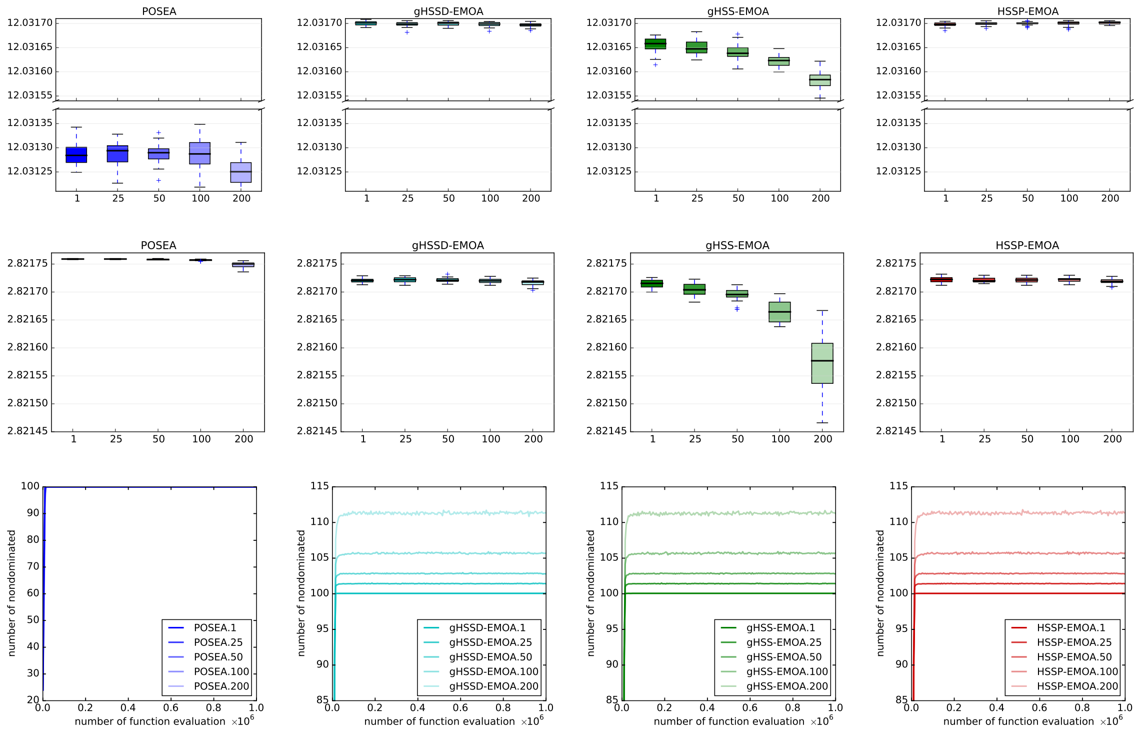


Figure 6.3: Summarized results from 30 runs of POSEA and of gHSSD-, gHSS- and HSSP-EMOAs for DTLZ2convex in $d = 2$ (from left to right column, respectively).

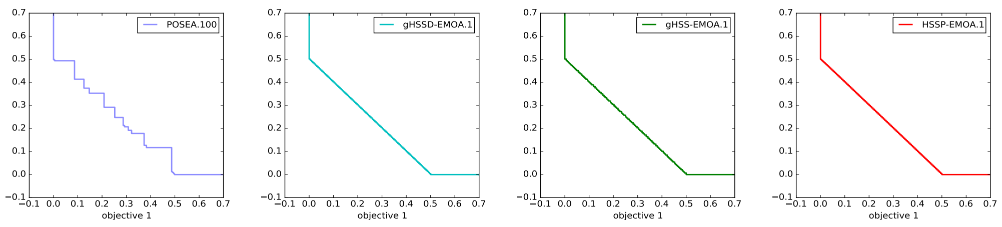


Figure 6.4: Final populations for DTLZ1 in $d = 2$.

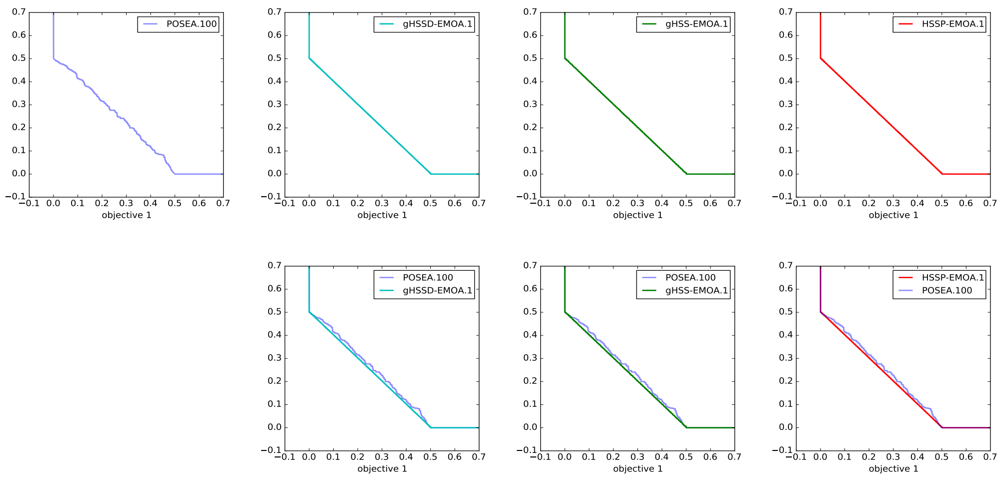


Figure 6.5: The 50% EAFs for DTLZ1 in $d = 2$.

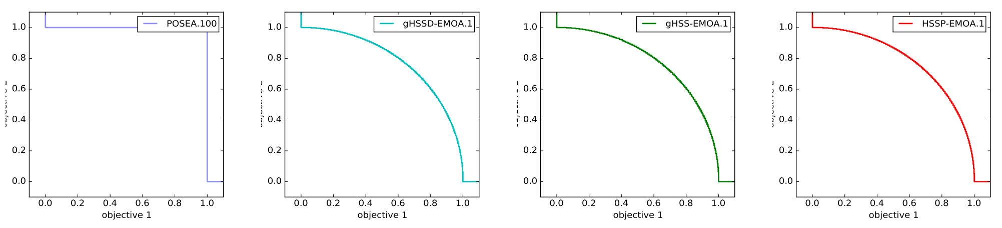


Figure 6.6: Final populations for DTLZ2 in $d = 2$.

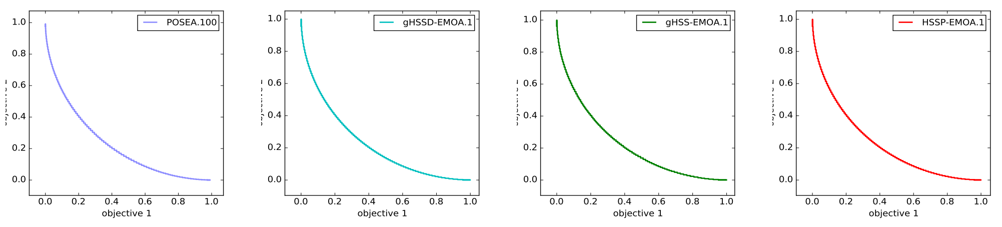


Figure 6.7: Final populations for DTLZ2convex in $d = 2$.

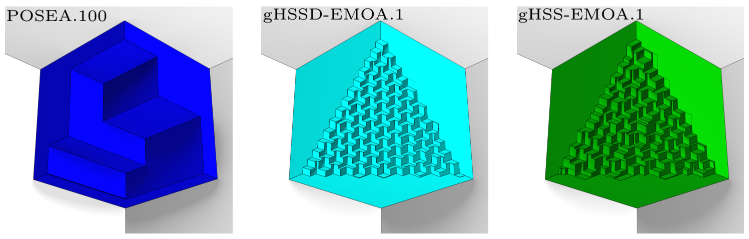


Figure 6.8: Final populations for DTLZ1 in $d = 3$.

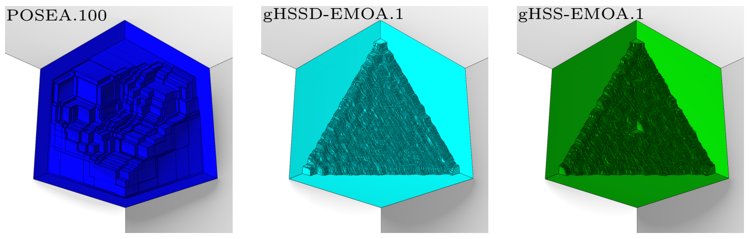


Figure 6.9: The 50% EAFs for DTLZ1 in $d = 3$.

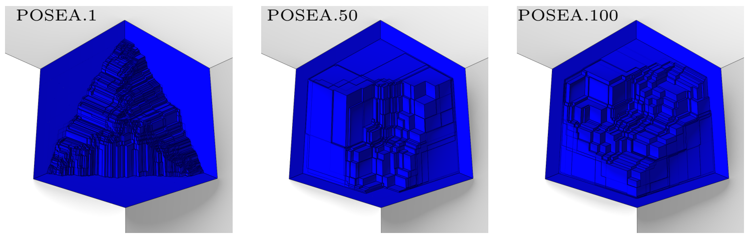


Figure 6.10: The 50% EAFs for POSEA on DTLZ1 in $d = 3$.

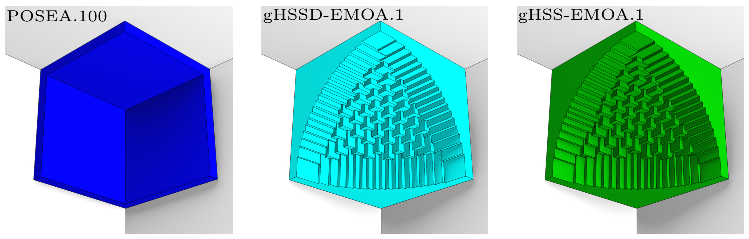


Figure 6.11: Final populations for DTLZ2 in $d = 3$.

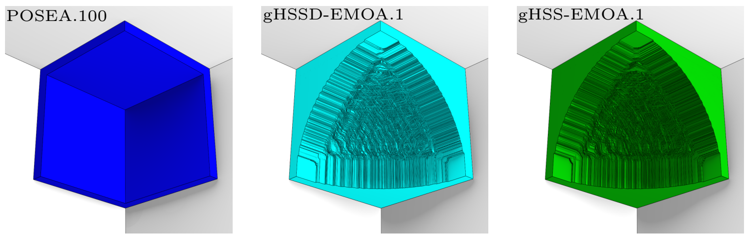


Figure 6.12: The 50% EAFs for DTLZ2 in $d = 3$.

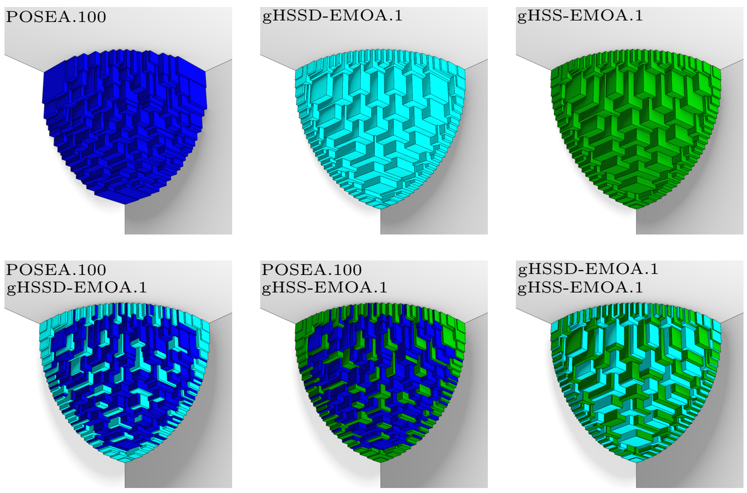


Figure 6.13: Final populations for DTLZ2convex in $d = 3$.

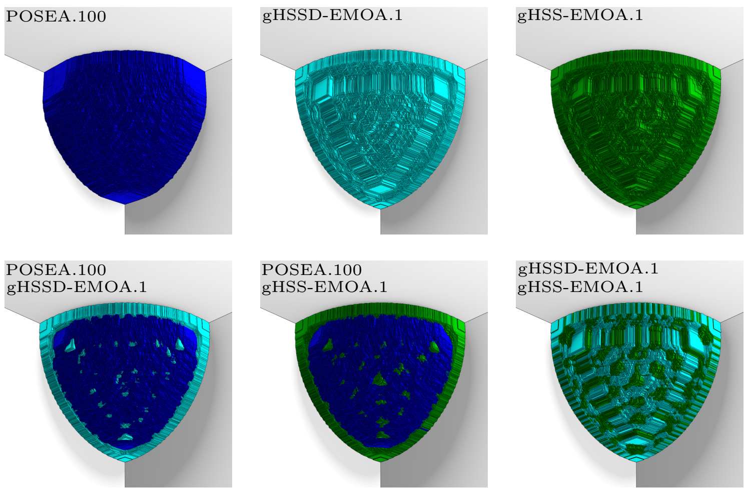


Figure 6.14: The 50% EAFs for DTLZ2convex in $d = 3$.

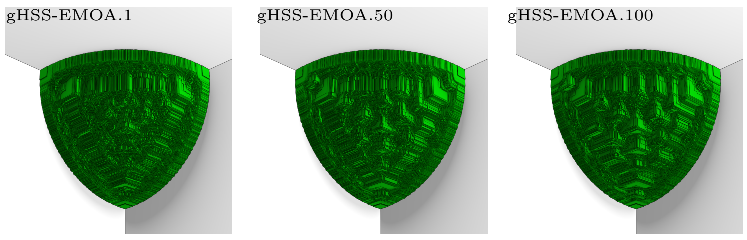
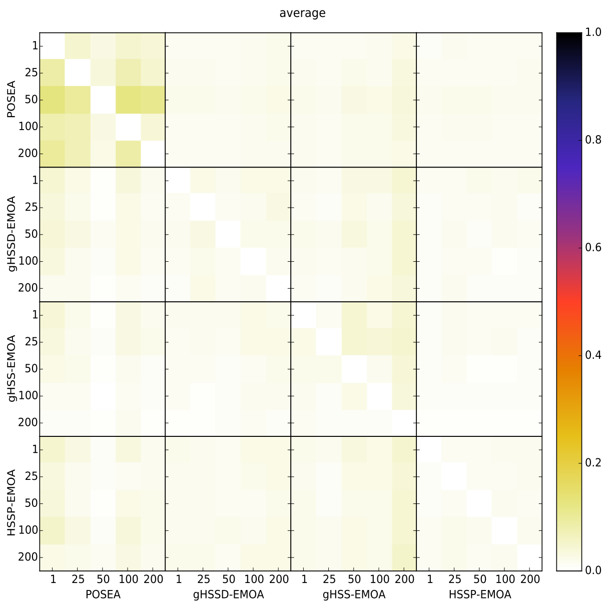
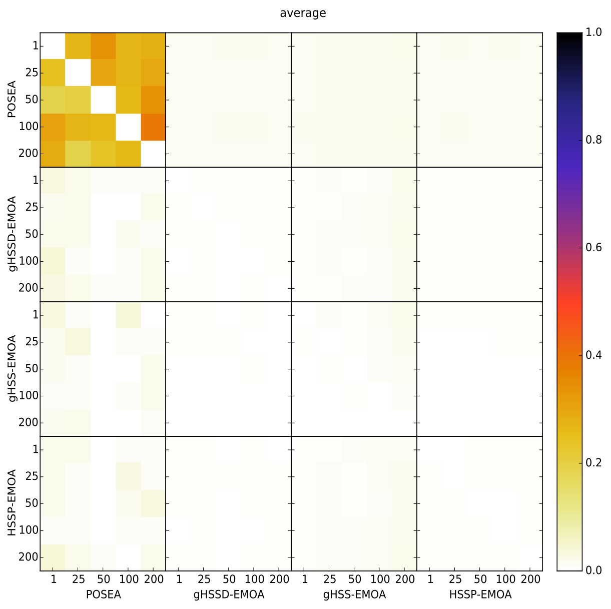


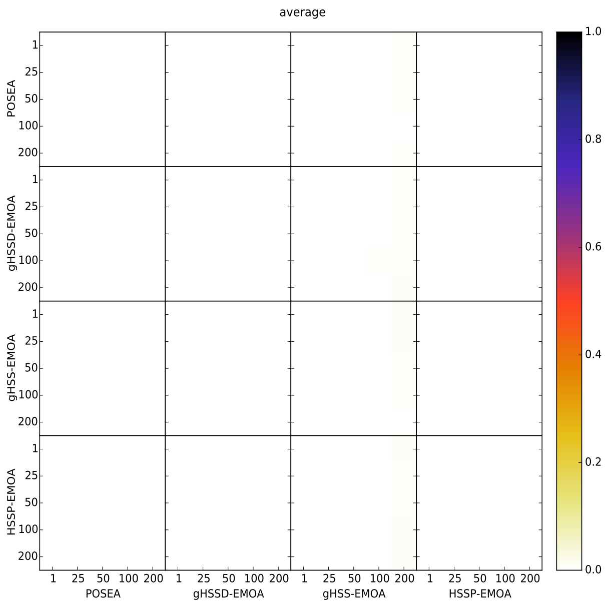
Figure 6.15: The 50% EAFs for gHSS-EMOA on DTLZ2convex in $d = 3$.



(a) DTLZ1



(b) DTLZ2



(c) DTLZ2convex

Figure 6.16: Each cell shows the average percentage of nondominated solutions in the final population of an EMOA (column) with $\lambda \in \{1, 25, 50, 100, 200\}$ which are strictly dominated by the final population of another EMOA (row) with a given $\lambda \in \{1, 25, 50, 100, 200\}$, for the DTLZ problems in $d = 2$.

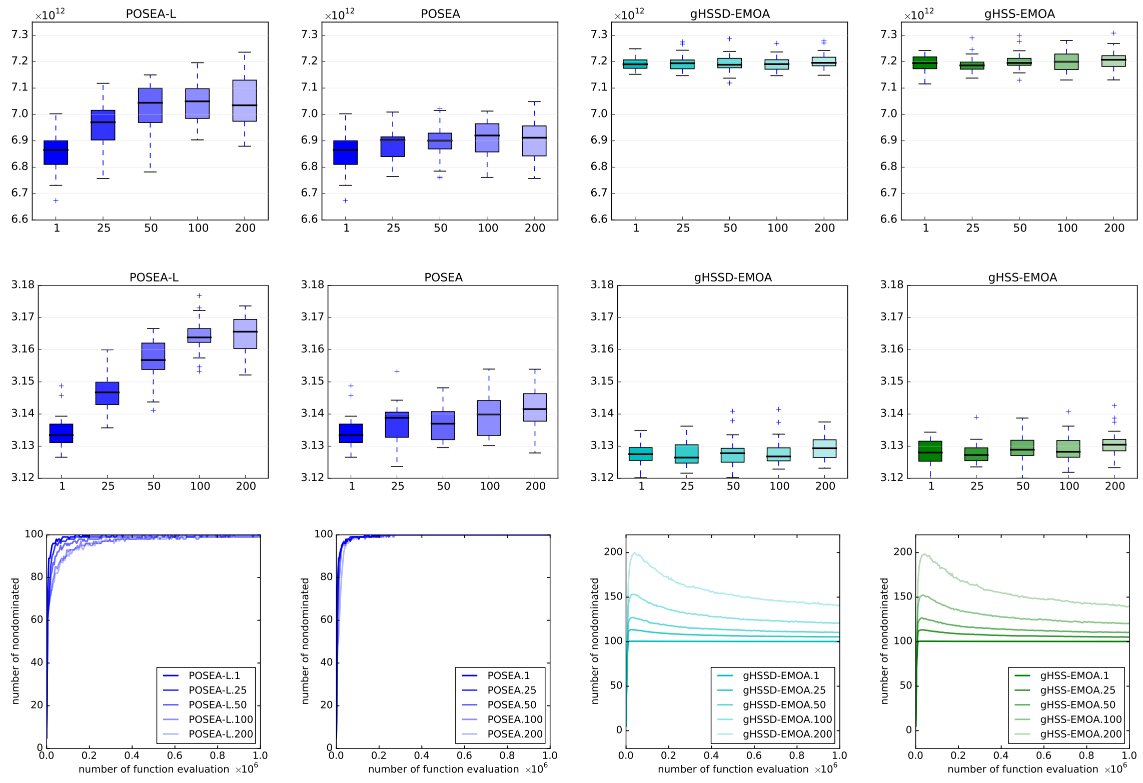


Figure 6.17: Summarized results from 30 runs of POSEA and of gHSSD- and gHSS-EMOAs for the multiobjective knapsack problem in $d = 3$ with 500 items (from left to right column, respectively).

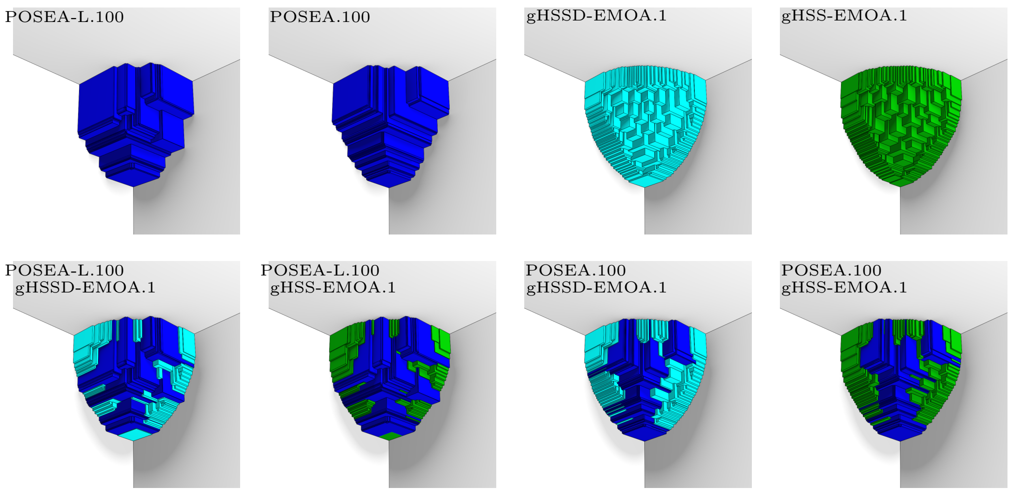


Figure 6.18: Final populations for the multiobjective knapsack problem in $d = 3$ with 500 items.

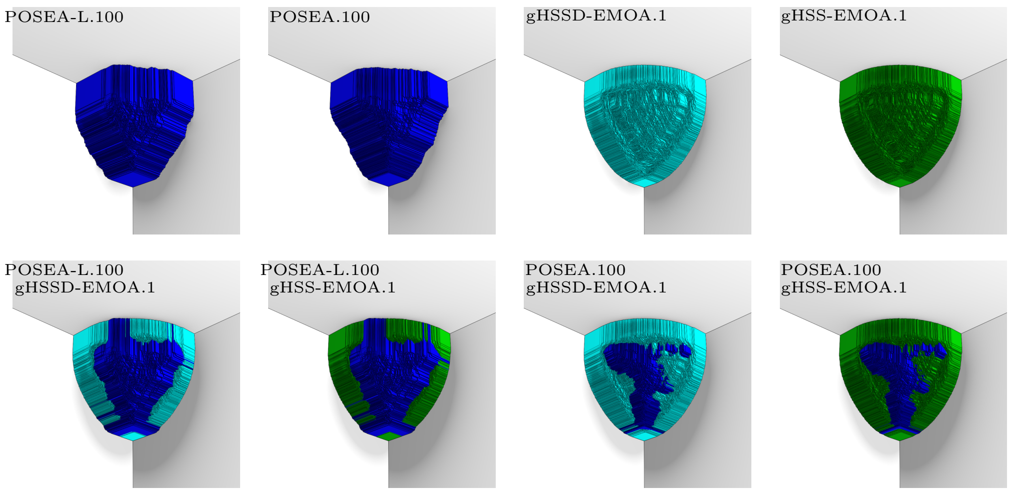
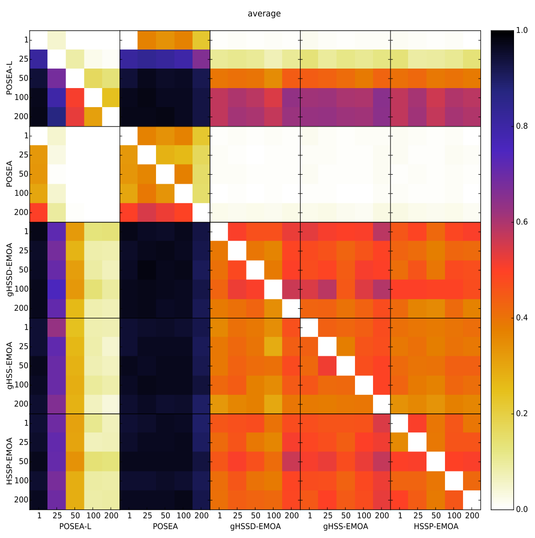
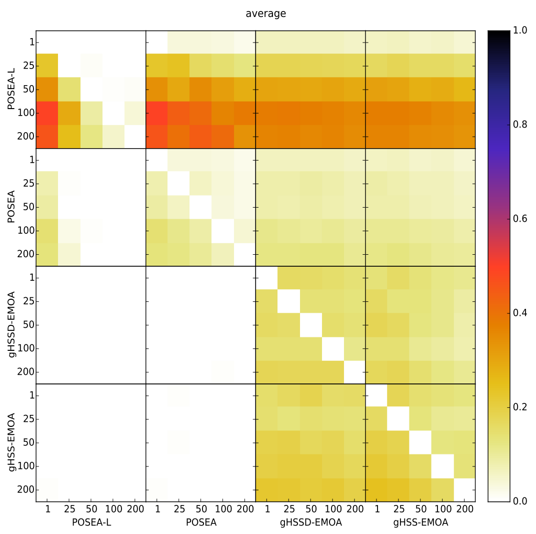


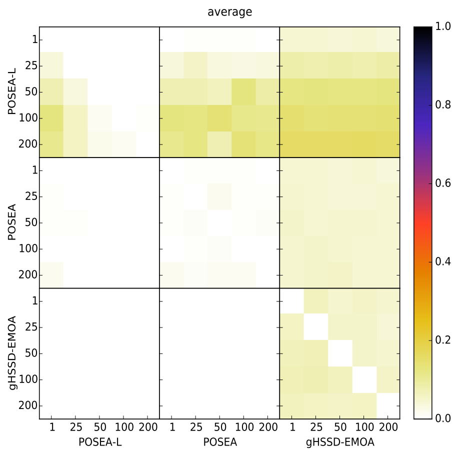
Figure 6.19: 50% EAFs for the multiobjective knapsack problem in $d = 3$ with 500 items.



(a) 2D



(b) 3D



(c) 4D

Figure 6.20: Average percentage of dominated solutions for the multiobjective knapsack problem with 500 items.

6.3.3 Hypervolume-Based EMOAs

HSSP-EMOA: Steady-state vs Generational

Although the generational environmental selection ($\lambda = \mu$) based on the exact computation of the HSSP currently has theoretical advantages (is effective) over the steady-state one ($\lambda = 1$), the results in Figures 6.1 to 6.3 do not show a clear difference in their performance. In fact, the average hypervolume indicator values are close for all values of λ, although a tendency for improvement was observable for increasing values of λ in the deterministic simulations in Section 4.5.1. The EMOA dynamics could be one of the reasons for this observation. Now the population moves towards the Pareto front and the mutation operator influences the searched space which will likely lead to different (sequences of) generated solutions for different settings of λ. Despite this, the main reason for the observed results may have to do with the actual number of nondominated solutions which the environmental selection can choose from at each generation. The last rows of Figures 6.1 to 6.3 show that, independently of the λ setting, the environmental selection operator has to choose 100 nondominated solution from a set, in average, of at most 115. These observations indicate that even though the EMOA is a generational EMOA from the parameter setting point-of-view, in practice it is far from being generational. In practice, the observed performance of HSSP-EMOA is comparable to that of HSSP-EMOA in Section 4.5.1 with a setting of λ between 1 and $\mu/4$.

Greedy EMOAs: Increasing λ

Similarly to HSSP-EMOA, there is no clear (dis)advantage of a specific setting of λ over the others concerning the average hypervolume indicator values obtained with gHSSD-EMOA, which contradicts the results from Section 4.5.1. The reason for this may also have to due with the introduction of stochastic behavior and also with the number of nondominated solutions to choose from at the environmental selection stage. Only with gHSS-EMOA and only in a few continuous problem instances (DTLZ2 and DTLZ2convex in $d = 2, 3$), is a decrease of hypervolume values observed as λ increases. A justification for observing this for gHSS-EMOA but not for gHSSD-EMOA may have to do with a possibly lower sensitivity of gHSS to stochastic aspects due to its choice consistency, i.e., the algorithm selects solutions in the same way as in the previous generation (chooses solutions from the parent population in the same order) until an offspring is selected. Even if the selected solutions are not the same as in the previous generation, it is possible that they are close in objective space. This consistency of gHSS is somewhat observable in the central region of the 50% EAF for DTLZ1 (see Figure 6.9), and even more for the 50% EAF for DTLZ2convex and as λ increases (see Figure 6.15). The prominent regions show a preference towards some regions of the objective space.

Incremental greedy vs decremental greedy

For the continuous problems, gHSS-EMOA with $\lambda = 1$ was, in general, competitive with gHSSD-EMOA with any setting of λ. With increasing values of λ for gHSS-EMOA there is a tendency to become outperformed by gHSSD-EMOA. However, for the multiobjective knapsack problem, both of them are competitive for any setting of λ.

The pairwise comparisons between the two greedy approaches in the dominance tables (see Figure 6.16 for example) do not show considerable differences in the average percentage of dominated solutions in the final population. This indicates that the differences in hypervolume values may not have to do so much with proximity to the Pareto front, but with different distribution of the final populations. In fact, the comparisons between the final populations and the 50% EAFs in Figures 6.8 to 6.14 show that the algorithms seem to complement each other, by finding solutions in the spaces left by one another.

There is no visible difference between the final populations and the 50% EAFs obtained by gHSS-EMOA and gHSSD-EMOA in two-dimensional instances (see Figures 6.4 to 6.7, for example), so let us look only at the three-dimensional ones. In the case of DTLZ1, it is visible in the example of Figure 6.8, that gHSSD-EMOA was able to retain a population that is very much uniformly distributed. Although gHSS-EMOA also retains solutions that are more or less well distributed, they do not look as uniform. This is not unexpected as in this case ($\lambda = 1$) gHSSD selects optimally and gHSS may not. For the DTLZ2, they both seem to obtain very similar distributions (see Figures 6.11 and 6.12). Note that they are well distributed in the center and in the boundary while the gap between the center and the boundary has to do with having the reference point set too far away. In the DTLZ2convex case (see Figures 6.13 and 6.14), the populations also seem equally well distributed and have many solutions in the boundary because the reference point is far away. The same was observed for the $\lambda = 100$ cases (not shown). Similar observations to this later case can be made concerning the final populations obtained for the multiobjective knapsack problem (see Figures 6.18 and 6.19). Note that as the front is not continuous, it is unlikely that in this case a more uniform distribution can be obtained.

Greedy vs Exact HSSP computation

The results for the $d = 2$ case observed in Figures 6.1 to 6.3 show that gHSSD-EMOA and also gHSS-EMOA (particularly for $\lambda = 1$) are generally competitive with HSSP-EMOA regarding the hypervolume indicator of the final populations. Looking at the distribution of the final populations and at the 50% EAF obtained with the three EMOAs (see Figures 6.4 to 6.7), there is no visible difference between them. Although one cannot be sure that this will generalize for the $d > 2$, and for all problems, these results are encouraging. The greedy algorithms could be a good alternative to expensive exact algorithms to perform environmental selection.

Hypervolume Indicator vs HSR indicator

The paper by Yevseyeva *et al.* [133] evidences that an increase of the hypervolume indicator of SMS-EMOAs population throughout the evolution is typically accompanied by an increase of the the HSR indicator. The same was observed in these experiments for all hypervolume-based EMOAs and thus, the plots were omitted. This behavior is expected as both indicators are strictly $\prec\cdot$-monotonic and thus, during the run and particularly in the beginning, it is expected that the current population dominates the previous one, and in such cases none of the two indicators will deteriorate. However, recall that maximizing the hypervolume indicator with a given number of points does not imply the maximization of HSR indicator nor vice-versa, as discussed in Section 5.5. Concerning increasing values of λ, if the average hypervolume of an EMOA's final population is visibly decreasing/increasing, typically so is the HSR indicator (for example, see Figures 6.1 to 6.3).

6.3.4 Sharpe-Ratio-Based EMOA

POSEA uses the HSR indicator for environmental selection and fitness assignment. For the DTLZ problems, the HSR indicator is used without modification while for the multiobjective knapsack problem, the Pareto dominance relation in the original HSR indicator is replaced by the preferability relation to account for infeasible solutions under a single model. Even though the latter is also Pareto dominance compliant, it is not immediately clear whether it also inherits the properties of the former. Thus, only the results on the DTLZ problems can be directly compared to the theoretical and experimental results from the previous chapter. For this reason, the results are discussed separately for each type of problem, first for the DTLZ problems and then, in the last part of this section, for the multiobjective knapsack problem.

Increasing λ

In general, the setting of λ does not seem to impact the quality of POSEA final populations regarding the HSR indicator and also the hypervolume indicator. See in Figures 6.1 to 6.3 that there is no clear advantage of a setting of λ over the others. Moreover, no clear differences are observable in the dominance tables either (see Figure 6.16). However, this does not imply that the outcome distributions will be similar. Although there is no clear difference in the 50% EAFs obtained by POSEA on most problem instances, those obtained for the linear fronts in $d = 3$ are visibly different (see Figure 6.10).

Hypervolume indicator vs HSR indicator

Similarly to the hypervolume-based EMOAs, but now with the roles of the two indicators exchanged, the increase of the HSR indicator (the indicator being maximized) typically lead to an increase of the hypervolume indicator during the run. Nevertheless, it may happen that a loss of hypervolume occurs along the run (this was observed only for the DTLZ2convex front in $d = 3$).

Final population distribution: DTLZ1

Given the theoretical results in Section 5.4.3 for the $d = 2$ linear front, the convergence of POSEA towards the optimal μ-distribution was expected, i.e., towards a set of μ points uniformly distributed on the Pareto front, including the two extreme points. It was also expected that as the population moves towards the Pareto front, and away from the reference point, the investment assigned to the extreme points would increase.

Figure 6.1 shows that the average number of nondominated solutions in the population along a run of POSEA is lower than μ (but becomes closer to μ as the number of dimensions increase). In the particular case of $d = 2$, the final population has, in average, less than 70 nondominated solutions. Figure 6.4 suggests that most of such nondominated solutions are located near the extremes. The 50% EAF in

Figure 6.5 confirms it, where the extremes of the EAF surface are closer to the Pareto front than the rest of the surface. Although a convergence to sets more uniformly distributed was expected, these results are not completely surprising. This is likely a consequence of setting the reference point too far. For example, assume that the populations contains only optimal solutions, including the extremes (points $(0, 0.5)$ and $(0.5, 0)$), and recall that $l = (0, 0)$ and $u = (1125, 1125)$. Then, by Theorem 5.11, the investment assigned to each extreme point would be, at least, 0.49977 while an inner point would be assigned, at most, $5e^{-4}$. Moreover, the experiments in Section 5.5.2 (in particular, see Figure 5.8) tell that if a non-extreme point is in a concave region, its investment will decrease to zero the closer it is to being dominated. The results observed for $d = 2$ seem to generalize to $d = 3$ (see Figures 6.8 and 6.10). In this case, it is more evident that POSEA focuses more on the extremes.

To conclude, the results suggest that the upper reference point is affecting POSEA's ability to find solution on the Pareto front. On the one hand, because of such a point, POSEA is likely to be strongly investing in the extreme points, and using mainly such points for parental selection. This will likely lead to generating many solutions close to the extremes, which makes the population move towards the front more quickly near the extremes than in the rest of the front. On the other hand, the investment in points between the extremes is either zero or very close to zero which leads to difficulties in maintaining such points (with zero investment they are likely to be discarded), and in generating points in the interior region of the front.

Final population distribution: DTLZ2

The results for the DTLZ2 front are as expected. The experiments in Section 5.5 for the $d = 2$ concave front already shown that only the extreme points are assigned strictly positive investment when the upper reference point is slightly far from the front. It is reasonable to expected a similar result for the $d = 3$ case. Consequently, POSEA obtains a final population that is in agreement with the expected results, i.e., with the preferences it expresses. In this case, the average number of nondominated solutions in the population is very low, at most 4 in the $d = 2$ case (see Figure 6.2) and at most 7 and 20 in the $d = 3$ and $d = 4$ cases, respectively (omitted plots). Note that any solution close to the extreme points, as it is in a concave region, will mostly likely get zero investment. Therefore, the low number of nondominated solutions is a consequence of having only a few solutions that have positive investment, and environmental selection will retain these few solutions while the remaining ones are selected arbitrarily to fill the population.

Final population distribution: DTLZ2convex

The results from Section 5.5 showed that the HSR indicator is interested in convex regions. Thus, it was expected that for a convex front POSEA would be able to find a good distribution as it is the case in the results shown in Figures 6.7 and 6.13. Note that the points in the 50% EAF in Figure 6.14 are very well distributed in the sense that there are no gaps nor prominent regions, i.e., the EAF retains the convex shape. Only for the convex fronts, POSEA has a population filled with nondominated solutions as all points in the front have positive investment and thus, environmental selection tries to keep as many nondominated solutions as possible.

Hypervolume-based vs HSR-based EMOAs

One of the main differences between POSEA and hypervolume-based EMOAs is that the latter fill the population with as many nondominated solutions as possible, which is a consequence of the strict $\prec$-monotonicity of the hypervolume indicator. The same is not ensured by the strict $\prec\cdot$-monotonicity of the HSR-indicator, and thus, it is possible for some nondominated solutions to be considered irrelevant from the indicator's point of view, and be assigned zero investment. This seems to happen, resulting in cases where only a few nondominated solutions survive and/or are given enough investment (fitness) to have a chance of reproduction. This leads to a more focused/greedy search by POSEA towards the regions that are more interesting according to its underlying preferences. The EMOAs also differ in how they are affected by the (upper) reference point. If set too far away, for linear and concave fronts POSEA becomes more interested in their extremes, while hypervolume-based EMOAs become more interested in boundary points if the front is convex or concave.

Even though the final populations found by POSEA for DTLZ1 did not contain as many points and as well uniformly distributed as hypervolume-based EMOAs, their average hypervolume indicator was very close, and their average HSR indicator was consistently better with POSEA than with hypervolume-based EMOAs. An explanation for this is that POSEA may be finding points slightly closer to the extremes of the Pareto front, and as the reference point is far away, any slight improvement is reflected in the indicator values. This is more clearly observed in the case of DTLZ2, where POSEA achieves better HSR indicator values, although the final population of POSEA contains very few nondominated solutions. The dominance table in Figure 6.16 shows that a final population of POSEA dominates, in average, approximately 5% of the nondominated solutions in a final population of hypervolume-based EMOAs (i.e., in average, dominates 5 out of 100 solutions), and vice-versa. Note that, as POSEA finds very few nondominated solutions then, if one solution was dominated in average, that would be around $30\% - 50\%$ of them in the $d = 2$ case (and $10\% - 20\%$ in the $d = 3$ case). This indicates that the few nondominated solutions found by POSEA rarely are dominated by those found by hypervolume-based EMOAs. Thus POSEA can, indeed, obtain solutions closer to the Pareto front extremes than hypervolume-based EMOAs.

For both concave and convex fronts (DTLZ2 and DTLZ2convex), the final populations obtained by POSEA consistently have lower hypervolume indicator than those by hypervolume-based EMOAs but, on the other hand, they consistently have higher HSR indicator. The gap between the indicator values obtained with both types of EMOAs accentuates as the number of dimensions increase. This clearly shows that maximizing one indicator is not the same as maximizing the other when the fronts are not linear. It is clear from the dominance tables and from the monotonicity properties of the indicators, that these differences are a reflection of finding different point distributions, and not of one EMOA finding a population dominating the population found by the other. For example, for the $d = 3$ convex front, the HSR indicator is less biased towards the boundary which allows the population in POSEA to get closer to the Pareto front altogether while gHSS-EMOA and gHSSD-EMOA concentrate a lot of points in the boundary (see Figures 6.13 and 6.14). Each EMOA is, as expected, biased towards the distributions and/or regions in objective space that better fulfill the indicator preferences.

The Multiobjective Knapsack Problem

Recall that, for the multiobjective knapsack problem, POSEA uses the PHSR indicator, which is a version of the HSR indicator whose underlying preference model was replaced to consider the preferability relation, so that it better accommodates infeasible solutions (see Section 5.7 for more details). Although the properties of the HSR indicator are not clearly inherited by the PHSR indicator in spite of their similarities, it would not be surprising if PHSR indicator exhibited identical preferences regarding feasible solutions.

It is reasonable to expect the Pareto front of the multiobjective knapsack problem instances to contain, loosely speaking, both convex and concave regions, and to see POSEA invest mostly in the former. From the optimization strategy point of view, it may be desirable to avoid investing too much in the convex regions, and to avoid having only a few solutions with strictly positive investment as it was observed for DTLZ1 and DTLZ2 problems. Thus, let us consider also the alternative POSEA-L that limits the maximum investment so that selection pressure is adjusted. POSEA-L forces the spread of investment and consequently, reduces the selective pressure. As referred in the original paper by Yevseyeva *et al.* [133], this imposition on investment may give rise to an investment strategy that can be viewed as disruptive selection, where some investment is assigned to dominated solutions, and which may benefit evolution.

Although in the experiments of Figure 6.17 both POSEA and POSEA-L use the PHSR indicator, the HSR indicator is still used to evaluate the final populations. In this case, the investment constraint is beneficial both regarding the hypervolume and the HSR indicator values of the final population. With POSEA-L, there is a clear difference in its performance with different settings of λ, the greater λ is, the better, at least up to $\lambda = \mu$. Note that the investment constraint forces POSEA-L to invest in, at least, λ solutions when $\lambda < \mu$ and in μ solutions otherwise. Having at least μ solutions with strictly positive investment (possibly equally distributed) when $\lambda \geq \mu$, is helpful to the environmental selection. Otherwise, the population has to be filled with arbitrarily selected solutions. It is possible that the performance improvement is more related to this fact, that μ solutions are explicitly chosen with the maximization of the PHSR indicator, and not so much with the setting of λ.

Figure 6.17 shows that POSEA-L outperforms POSEA regarding both indicators. Similarly to the results on DTLZ fronts, in comparison to hypervolume-based EMOAs, the Sharpe-ratio based EMOAs consistently obtained better results on the HSR indicator, and worse results regarding the hypervolume indicator. In Figure 6.18, the final population of the first run suggests that the Pareto front has a convex shape even though it might have some locally concave regions. POSEA and POSEA-L cannot find very well distributed solutions, but it is evident that they can approximate the Pareto front better in the inner region, whereas hypervolume-based EMOAs are better at finding solutions in the boundary (similarly to DTLZ2convex). POSEA-L seems to obtain slightly better distributions of solutions than POSEA, which is observed more clearly in the 50% EAFs (see Figure 6.19). Actually, POSEA-L approximates the behavior of POSEA in a continuous convex front. In this case, POSEA-L can better approximate the Pareto front in the center and hypervolume-based EMOAs approximate better in the boundary as can be observed in Figures 6.18 and 6.19.

In general, POSEA-L can approximate the Pareto front better than hypervolume-based EMOAs. See in Figure 6.20 that the average number of solutions in the final population of hypervolume-based EMOAs that are dominated by the final population found by POSEA-L (with $\lambda \geq \mu$) is approximately, $50\% - 60\%$ in $d = 2$ case, 40% in $d = 3$ case, and 20% in $d = 4$ case. On the other hand, almost none of the solutions found by POSEA-L are dominated by the final populations found by hypervolume-based EMOAs. This may be the reason why POSEA-L consistently obtains better HSR indicator and worse hypervolume indicator than hypervolume-based EMOAs. The former seems to value more the proximity to the Pareto front while the latter seems to value more a well spread distribution slightly away from it.

6.4 Concluding Remarks

The first main result of this chapter is the validation of POSEA and, in particular, of the use of the Sharpe-ratio indicator for both environmental selection and fitness assignment. Moreover, the experiments allow to understand POSEA and get some insight on what to expect from it. They also allow to understand that the Sharpe ratio's underlying model can be easily modified to help enhance the optimization search process, for example, by introducing constraints limiting the investment in a single solution. The second main result is the validation of greedy algorithms for environmental selection as a way of approximating the HSSP and replacing the expensive computation required to solve it exactly. The results showed that greedy-based EMOAs are competitive with an EMOA based on exact solutions to the HSSP with any setting of λ in the $d = 2$ case and, in $d = 3, 4$ are competitive with HSSP-EMOA with $\lambda = 1$ (equivalent to SMS-EMOA and gHSSD-EMOA with $\lambda = 1$).

The fact that POSEA may tend to disregard solutions in concave regions and, sometimes, to only focus on a few solutions (in the convex regions) should not be interpreted as bad performance. It is only a reflection of its underlying preferences. Although the upper reference point for HSR indicator can be used to state a preference for extreme solutions, in these experiments it was set far away only to guarantee that all feasible solutions dominate it. However, it might be the case that the DM is not more interested in an extreme solution than it is in any other nondominated solutions. The upper reference point could be adjusted along the run. However, that could change the ranking of sets and consequently the search process. If the HSR indicator does not reflect the DM preferences given some set of points, then an alternative is to change the underlying model and take advantage of the flexibility of the class of Sharpe Ratio indicators.

Finally, the results show that each indicator-based EMOA seeks subsets of solutions that are the best according to its indicator, and thus frequently one finds solutions that are not attained by another EMOA and vice-versa. This chapter shows that, in such cases, one cannot generally say that a particular EMOA is better than another as, in the end, what matters is that DM preferences are satisfied. If such preferences are not known then there is no guarantee that the best choice is to maximize the hypervolume indicator or another indicator. It is important however, to understand the preferences underlying an indicator so that one can say that some EMOA is better at fulfilling these or those preferences based on how well it maximizes the indicator.

Chapter 7

Conclusion

The focus of this thesis was the selection step in Evolutionary Algorithms (EAs) from the more general perspective of Portfolio Selection. Throughout the last decades, the perspective on selection in EAs for multiobjective optimization problems has evolved. Initially, the focus was mainly on the solution (individual) quality but soon the importance of having diverse solutions in the population led to the inclusion of diversity preservation techniques. Later on, the use of quality indicators became a successful means of selecting sets of individuals accounting both for individual solution quality and diversity in the selected set in an integrated way. With such approaches, the focus changed to the quality of the set of selected solutions. Until recently, environmental and fitness assignment have been mostly viewed as two separate problems in EAs. The more recent approach that looks at selection from a Portfolio Selection perspective includes fitness assignment as part of the selection problem and sheds new light on the integration of preferences.

An increasing concern with the theoretical properties of selection methods has accompanied the change of perspective over selection in EAs. The current view of environmental selection as a subset selection problem, formulated as the search for a subset maximizing a quality indicator, is strengthened by the use of theoretically backed up quality indicators that led to equipping indicator-based EMOAs with theoretical support. Note that, for example, in the unknown DM preferences scenario, although many solutions are incomparable, selection methods in EMOAs will inevitable have to choose some solutions over other incomparable ones. The theoretical characterization of a quality indicator allows one to understand the biases (the inner preferences) and possible limitations introduced by selection methods based on such an indicator and, consequently, learn about what to expect from the outcome of the resulting indicator-based EMOA. Understanding the theoretical properties granted by selection methods became an important aspect in EMOAs.

There are, at least, two possible paths to improve upon the state-of-the-art selection methods aligned with the concern for theoretically supported selection mechanisms, and this thesis pursued both of these paths. One is to solve the obstacles of known theoretically-strong selection methods, the other is to theoretically study new approaches to selection in EMOAs. In line with these two paths, the contributions of this thesis can be grouped into two major contributions, each one with different impacts. The first major contribution has immediate practical implications and consists of lowering the computational barriers to hypervolume-based selection. The second major contribution, and perhaps the one with greater implications for the future of selection in EAs, concerns the study of selection from the point of view of a Portfolio Selection Problem (through a particular formulation based on the Sharpe ratio). To complement these major contributions, a study on the impact of such selection methods on the outcome quality of EMOAs was provided.

7.1 Hypervolume-Based Selection

The theoretical properties of the hypervolume indicator motivated the development of several new algorithms for low dimensional cases of hypervolume-based problems (in Chapter 4) matching or improving upon the practical and asymptotic performance of the state-of-the-art algorithms. As the ultimate goal was to efficiently solve the Hypervolume Subset Selection Problem (HSSP) in the context of EMOAs, the first step was to exploit the behavior of steady-state hypervolume-based EMOAs, in particular, of SMS-EMOA. Thus, at the core of the new algorithms is the update of previously computed solutions. In particular, an algorithm to update the HSSP with $k = n - 1$ for the $d = 3$ case in linear time was developed, which was based on the update of hypervolume contributions. This algorithm allowed to take advantage of results known from previous generations in SMS-EMOA. Because of the intrinsic relation between the several hypervolume-related problems (discussed in Chapter 3), the application of such update algorithm goes beyond the efficient update of SMS-EMOA populations and of the solution of HSSP given $k = n - 1$. For example, it boosted the following work on greedy algorithms to approximate the HSSP.

New algorithms for 3 and/or 4 dimensions were proposed using one of two greedy approaches, an incremental and a decremental one, supported on the update of hypervolume contributions. To strengthen the adequacy of such approaches as alternatives to the currently very expensive algorithms for the exact computation of the HSSP, this thesis narrowed down the knowledge gap on their approximation guarantees by providing approximation ratios to the optimal solution. Together with the experiments in Chapter 6 these approximation algorithms were shown to provide a good and computationally cheaper alternative to the exact computation of HSSP. Overall, the work developed resulted in a set of algorithms that contribute to making hypervolume-based selection more affordable while still providing theoretical support.

7.2 Portfolio-Based Selection

The positive results [133] obtained in the first study of EA selection formulated as a portfolio selection problem, solved through the Sharpe ratio maximization, provided the motivation for a deeper study of such a formulation. This thesis focused mainly on studying whether such a formulation can also be theoretically supported and on exploring some of its potential advantages. The first step was to formalize a new class of indicators, the Sharpe ratio indicator class. The indicator in [133] was formalized as an instance of this class, the HSR indicator. This thesis presented the first analytical and numerical studies on the characterization of such an indicator. The indicator is shown to have monotonicity properties, and to be scaling invariant under linear transformations. The optimal μ-distributions on a 2-dimensional objective space were determined exactly for linear fronts, and were numerically approximated for other fronts (in Chapter 5). These results were complemented and validated with experimental results on EMOAs using such an indicator both for environmental selection and for fitness assignment (in Chapter 6). The theoretical and experimental results on the HSR indicator are a proof-of-concept of the portfolio selection view of EA selection. Two other instances of the Sharpe ratio indicator class were formalized, covering the preference uncertainty scenarios for two other classes of multiobjective optimization problems, one for constrained multiobjective optimization problems and another for set-valued optimization problems (in Chapter 6). These formulations unveil the flexibility of the HSR indicator class (and of the portfolio selection model, in general) to accept different preference models and, through these models, allow different types of problems to be contemplated without any modification to the selection method. The interesting exploratory results shown for these formulations motivate a more in-depth study as future work.

7.3 Future Work

The work developed in Chapter 5 on Portfolio-based selection opens up several directions for future work, of which (a combination of) the following can be outlined:

1. extend theoretical and numerical analysis to more than two dimensions and to other Sharpe ratio indicator instances;
2. integrate DM preference information over objective/decision space;
3. study extensions of the Sharpe ratio model to enhance preference-based search in EAs;
4. study portfolio selection formulations alternative to the Sharpe ratio-based one;
5. consider uncertainty in the objective functions.

The first direction for future work describes the obvious extensions to the work in Chapter 5. Regarding the second direction, although in this thesis all Sharpe ratio-based instances considered the scenario of unknown DM preferences, the scenarios where preference information of different types is available can be considered as well. Preferences can be integrated in different ways. One way is by replacing the dominance relation used in the formalization of HSR indicator by another one (as was done with, for example, the preferability relation). Another way is to consider that the goal vector expressing the DM preferences is drawn from a distribution other than the uniform one. It is also plausible to expect that MCDA methods for expressing preferences can be used as well. The integration of such information would allow Sharpe ratio-based indicators that are compliant with the DM preferences to be formalized, which could be used for assessing how well a set/portfolio satisfies the DM and/or for use in preference-based search in EAs.

The third future work direction is related to how the portfolio selection model could be extended to enhance the EA search process. For example, in Chapter 6, the integration of a constraint limiting the investment on each individual was beneficial to the EA search process for the combinatorial problem. Different constraints can be added, such as one defining that a given asset cannot be assigned more investment than another one. Concerning the fourth direction, different ways of balancing expected return and risk can be considered. For example, the Sharpe ratio could be replaced by other performance indexes, such as Sortino and Traynor ratios. That would lead to formulations possibly easier to compute (e.g., linear programming problem), and to use a different investment strategy thus leading to indicators with different characteristics.

Finally, the last direction of future work is to consider selection over multiobjective optimization problems subject to uncertain objective functions, whether or not in combination with the already considered uncertainty regarding DM preferences. This type of problems was only superficially discussed in Chapter 5 with the example of worst case robustness problems (see Section 5.6.1), and the work developed in Section 5.6.2 points out a possible direction for modeling the associated uncertainty. The PSP formulation of this type of problems is already gaining interest in the scientific community, where an alternative model considering both sources of uncertainty is being developed [94, Sect. 4.1].

7.4 Final Remarks

Overall, this thesis explored the selection at the core of EAs mostly from the multiobjective optimization perspective, but instead of interpreting it as just a subset selection problem, it explores the more general view as a portfolio selection problem. This thesis validates this view and lays the foundations for the formulation of other Sharpe-ratio indicator instances and for their analysis, and motivates other formulations of portfolio-based selection methods in EAs. This thesis explored a part of the potential of the formulation of environmental selection and fitness assignment as a Portfolio Selection Problem. The scope of portfolio selection in EAs is much wider, and this work opens up a new perspective on a formulation of selection that can be used for both known and unknown DM preference information, and for single and multiple objective optimization. The balance between individual quality and population diversity can be achieved with the Sharpe ratio or another method that adjusts better to the desired balance between the two. Finally, the impact of this work is not restricted to EAs. The problem at the core of this work is a decision making problem and thus, this work finds application, for example, in MCDA, to aid Decision Makers. To conclude, this thesis sheds new light on decision making and on selection in EAs, which can change the way in which preference information is integrated in the decision process.

References

[1] M. Alonso and L. Rodríguez-Marín. Optimality conditions for set-valued maps with set optimization. *Nonlinear Analysis: Theory, Methods & Applications*, 70(9):3057–3064, 2009.

[2] A. Auger, J. Bader, and D. Brockhoff. Theoretically investigating optimal μ-distributions for the hypervolume indicator: First results for three objectives. In R. Schaefer et al., editors, *Parallel Problem Solving from Nature — PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 586–596. Springer Berlin Heidelberg, 2010.

[3] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 563–570. ACM, 2009.

[4] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point. In *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms*, FOGA '09, pages 87–102. ACM, 2009.

[5] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science*, 425:75–103, Mar. 2012.

[6] G. Avigad and J. Branke. Embedded evolutionary multi-objective optimization for worst case robustness. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, pages 617–624. ACM, 2008.

[7] G. Avigad, E. Eisenstadt, and M. W. Cohen. Optimal strategies for multi objective games and their search by evolutionary multi objective optimization. In *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, pages 166–173, Aug. 2011.

[8] S. Azarm and J. Wu. Metrics for quality assessment of a multiobjective design optimization solution set. *Journal of Mechanical Design*, 123(1):18–25, 2001.

[9] J. Bader, D. Brockhoff, S. Welten, and E. Zitzler. On using populations of sets in multiobjective optimization. In M. Ehrgott et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 5467 of *Lecture Notes in Computer Science*, pages 140–154. Springer Berlin Heidelberg, 2009.

[10] J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, Spring 2011.

[11] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[12] M. Basseur, B. Derbel, A. Goëffon, and A. Liefooghe. Experiments on greedy and local search heuristics for d–dimensional hypervolume subset selection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '16, pages 541–548. ACM, 2016.

[13] J. L. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23:214–229, Apr. 1980.

[14] N. Beume. S-metric calculation by considering dominated hypervolume as Klee's measure problem. *Evolutionary Computation*, 17(4):477–492, Winter 2009.

[15] N. Beume, C. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, Oct. 2009.

[16] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[17] N. Beume and G. Rudolph. Faster S-metric calculation by considering dominated hypervolume as Klee's measure problem. *Proceedings 2nd IASTED Conference on Computational Intelligence*, pages 231–236, 2006.

[18] H.-G. Beyer and H.-P. Schwefel. Evolution strategies – A comprehensive introduction. *Natural Computing*, 1(1):3–52, Mar. 2002.

[19] L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle. Automatic component-wise design of multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(3):403–417, June 2016.

[20] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, New York, NY, USA, 2004.

[21] L. Bradstreet, L. Barone, and L. While. Maximising hypervolume for selection in multi-objective evolutionary algorithms. In *CEC 2006, IEEE Congress on Evolutionary Computation*, pages 1744–1751, July 2006.

[22] L. Bradstreet, L. Barone, and L. While. Updating exclusive hypervolume contributions cheaply. In *CEC 2009, IEEE Congress on Evolutionary Computation*, pages 538–544, May 2009.

[23] L. Bradstreet, L. While, and L. Barone. Incrementally maximising hypervolume for selection in multi-objective evolutionary algorithms. In *CEC 2007, IEEE Congress on Evolutionary Computation*, pages 3203–3210, Sept. 2007.

[24] L. Bradstreet, L. While, and L. Barone. A fast incremental hypervolume algorithm. *IEEE Transactions on Evolutionary Computation*, 12(6):714–723, Dec. 2008.

[25] L. Bradstreet, L. While, and L. Barone. Correction to “A fast incremental hypervolume algorithm”. *IEEE Transactions on Evolutionary Computation*, 13(5):1193–1193, Oct. 2009.

[26] L. Bradstreet, L. While, and L. Barone. A new way of calculating exact exclusive hypervolumes. *The University of Western Australia, School of Computer Science & Software Engineering, Technical Report UWACSSE-09-002*, 2009.

[27] L. Bradstreet, L. While, and L. Barone. A fast many-objective hypervolume algorithm using iterated incremental calculations. In *CEC 2010, IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[28] J. Branke. *MCDA and Multiobjective Evolutionary Algorithms*, pages 977–1008. Springer New York, 2016.

[29] K. Bringmann. Klee’s measure problem on fat boxes in time $O(n^{(d+2)/3})$. In *Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry*, SoCG ’10, pages 222–229. ACM, 2010.

[30] K. Bringmann. Bringing order to special cases of Klee’s measure problem. In K. Chatterjee and J. Sgall, editors, *Mathematical Foundations of Computer Science 2013*, volume 8087 of *Lecture Notes in Computer Science*, pages 207–218, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[31] K. Bringmann, S. Cabello, and M. T. M. Emmerich. Maximum volume subset selection for anchored boxes. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, pages 22:1–22:15, Schloss Dagstuhl Leibniz-Zentrum für Informatik, Germany, 2017. Dagstuhl Publishing.

[32] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. In S.-H. Hong et al., editors, *Algorithms and Computation*, volume 5369 of *Lecture Notes in Computer Science*, pages 436–447. Springer Berlin Heidelberg, 2008.

[33] K. Bringmann and T. Friedrich. Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. In M. Ehrgott et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 5467 of *Lecture Notes in Computer Science*, pages 6–20, Berlin, Heidelberg, 2009. Springer, Springer Berlin Heidelberg.

[34] K. Bringmann and T. Friedrich. Don’t be greedy when calculating hypervolume contributions. In *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms*, FOGA ’09, pages 103–112. ACM, 2009.

[35] K. Bringmann and T. Friedrich. An efficient algorithm for computing hypervolume contributions. *Evolutionary Computation*, 18(3):383–402, Fall 2010.

[36] K. Bringmann and T. Friedrich. Convergence of hypervolume-based archiving algorithms. *IEEE Transactions on Evolutionary Computation*, 18(5):643–657, Oct. 2014.

[37] K. Bringmann, T. Friedrich, and P. Klitzke. Two-dimensional subset selection for hypervolume and epsilon-indicator. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 589–596. ACM, 2014.

[38] K. Bringmann, T. Friedrich, and P. Klitzke. Efficient computation of two-dimensional solution sets maximizing the epsilon-indicator. In *CEC 2015, IEEE Congress on Evolutionary Computation*, pages 970–977, 2015.

[39] D. Brockhoff. *Many-Objective Optimization and Hypervolume Based Search.* PhD thesis, ETH Zurich, Switzerland, Aachen, Germany, 2009.

[40] D. Brockhoff. Optimal μ-distributions for the hypervolume indicator for problems with linear bi-objective fronts: Exact and exhaustive results. In K. Deb et al., editors, *Simulated Evolution and Learning*, volume 6457 of *Lecture Notes in Computer Science*, pages 24–34. Springer Berlin Heidelberg, 2010.

[41] D. Brockhoff, J. Bader, L. Thiele, and E. Zitzler. Directed multiobjective optimization based on the weighted hypervolume indicator. *Journal of Multi-Criteria Decision Analysis*, 20(5):291–317, July 2013.

[42] D. Brockhoff, T.-D. Tran, and N. Hansen. Benchmarking numerical multiobjective optimizers revisited. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 639–646. ACM, 2015.

[43] D. Brockhoff, T. Wagner, and H. Trautmann. On the properties of the R2 indicator. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 465–472. ACM, 2012.

[44] D. Brockhoff and E. Zitzler. Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In *CEC 2007, IEEE Congress on Evolutionary Computation*, pages 2086–2093, Sept. 2007.

[45] E. Carrano, G. Ribeiro, E. Cardoso, and R. Takahashi. Subpermutation based evolutionary multiobjective algorithm for load restoration in power distribution networks. *IEEE Transactions on Evolutionary Computation*, 20(4):546–562, Aug. 2016.

[46] T. M. Chan. A (slightly) faster algorithm for Klee's measure problem. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, SoCG '08, pages 94–100. ACM, 2008.

[47] T. M. Chan. Klee's measure problem made easy. In *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 410–419, Los Alamitos, CA, USA, 2013. IEEE Computer Society.

[48] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[49] G. Cornuejols and R. Tütüncü. *Optimization Methods in Finance.* Mathematics, Finance and Risk. Cambridge University Press, 2007.

[50] W. Cox and L. While. Improving and extending the HV4D algorithm for calculating hypervolume exactly. In B. H. Kang and Q. Bai, editors, *AI 2016: Advances in Artificial Intelligence*, volume 9992 of *Lecture Notes in Computer Science*, pages 243–254. Springer International Publishing, 2016.

[51] W. Cox and L. While. Improving the IWFG algorithm for calculating incremental hypervolume. In *CEC 2016, IEEE Congress on Evolutionary Computation*, pages 3969–3976, July 2016.

[52] K. Dächert and K. Klamroth. A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. *Journal of Global Optimization*, 61(4):643–676, Apr. 2015.

[53] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):181–197, Apr. 2002.

[54] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, volume 1, pages 825–830, May 2002.

[55] C. Doerr. Non-static parameter choices in evolutionary computation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 736–761. ACM, 2017.

[56] J. S. Dyer. *Multiattribute Utility Theory (MAUT)*, pages 285–314. Springer New York, New York, NY, 2016.

[57] M. Ehrgott. *Multicriteria Optimization.* Springer-Verlag Berlin Heidelberg, first edition, 2000.

[58] M. Ehrgott. *Multicriteria Optimization.* Springer-Verlag Berlin Heidelberg, second edition, 2005.

[59] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* Natural Computing. Springer, Oct. 2008.

[60] M. Emmerich, N. Beume, and B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In C. A. Coello Coello et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 62–76. Springer Berlin Heidelberg, 2005.

[61] M. T. Emmerich and C. M. Fonseca. Computing hypervolume contributions in low dimensions: Asymptotically optimal algorithm and complexity results. In *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 121–135. Springer Berlin Heidelberg, 2011.

[62] M. Fleischer. The measure of Pareto optima applications to multi-objective metaheuristics. In C. M. Fonseca et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 519–533. Springer Berlin Heidelberg, 2003.

[63] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution.* Wiley Publishing, 1966.

[64] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423. Morgan Kaufmann Publishers Inc., 1993.

[65] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

[66] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms I: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 28(1):26–37, Jan. 1998.

[67] C. M. Fonseca, A. P. Guerreiro, López-Ibáñez, and L. Paquete. On the computation of the empirical attainment function. In R. H. C. Takahashi et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 106–120. Springer Berlin Heidelberg, 2011.

[68] C. M. Fonseca, L. Paquete, and M. López-Ibáñez. An improved dimension-sweep algorithm for the hypervolume indicator. In *CEC 2006, IEEE Congress on Evolutionary Computation*, pages 1157–1163, July 2006.

[69] A. S. Fraser. Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science*, 10:484–491, 1957.

[70] T. Friedrich, K. Bringmann, T. Voß, and C. Igel. The logarithmic hypervolume indicator. In *Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms*, FOGA '11, pages 81–92. ACM, 2011.

[71] T. Friedrich and F. Neumann. Maximizing submodular functions under matroid constraints by multi-objective evolutionary algorithms. In T. Bartz-Beielstein et al., editors, *Parallel Problem Solving from Nature — PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 922–931. Springer International Publishing, 2014.

[72] T. Glasmachers. Optimized approximation sets for low-dimensional benchmark Pareto fronts. In T. Bartz-Beielstein et al., editors, *Parallel Problem Solving from Nature — PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 569–578. Springer International Publishing, 2014.

[73] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[74] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. J. Rawlins, editor, *FGA1*, volume 1, pages 69–93, 1991.

[75] R. J. Gomes. Branch-and-bound for the hypervolume subset selection problem. Master's thesis, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal, 2017.

[76] R. J. Gomes, A. P. Guerreiro, T. Kuhn, and L. Paquete. Implicit enumeration strategies for the hypervolume subset selection problem. *Computers & Operations Research*, 100:244–253, Dec. 2018.

[77] V. Grunert da Fonseca and C. M. Fonseca. The relationship between the covered fraction, completeness and hypervolume indicators. In J.-K. Hao et al., editors, *Artificial Evolution*, volume 7401 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin Heidelberg, 2012.

[78] A. P. Guerreiro. Efficient algorithms for the assessment of stochastic multiobjective optimizers. Master's thesis, IST, Technical University of Lisbon, Portugal, 2011.

[79] A. P. Guerreiro. gHSS-v1.1, Feb. 2017. DOI: `10.5281/zenodo.284559`.

[80] A. P. Guerreiro. HV4D-v1.2, Feb. 2017. DOI: `10.5281/zenodo.285214`.

[81] A. P. Guerreiro and C. M. Fonseca. Hypervolume Sharpe-ratio indicator: Theoretical results. Under review.

[82] A. P. Guerreiro and C. M. Fonseca. Hypervolume Sharpe ratio indicator: Formalization and first theoretical results. In J. Handl et al., editors, *Parallel Problem Solving from Nature — PPSN XIV*, volume 9921 of *Lecture Notes in Computer Science*, pages 814–823. Springer International Publishing, 2016. Best Paper Nominee.

[83] A. P. Guerreiro and C. M. Fonseca. Computing and updating hypervolume contributions in up to four dimensions. *IEEE Transactions on Evolutionary Computation*, 22(3):449–463, June 2018.

[84] A. P. Guerreiro, C. M. Fonseca, and M. T. M. Emmerich. A fast dimension-sweep algorithm for the hypervolume indicator in four dimensions. In *Canadian Conference on Computational Geometry (CCCG)*, pages 77–82, 2012.

[85] A. P. Guerreiro, C. M. Fonseca, and L. Paquete. Greedy hypervolume subset selection in the three-objective case. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 671–678. ACM, 2015. EMO Track Best Paper Award.

[86] A. P. Guerreiro, C. M. Fonseca, and L. Paquete. Greedy hypervolume subset selection in low dimensions. *Evolutionary Computation*, 24(3):521–544, Fall 2016.

[87] M. P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set, 1998.

[88] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[89] I. Hupkens and M. Emmerich. Logarithmic-time updates in SMS-EMOA and hypervolume-based archiving. In M. Emmerich et al., editors, *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*, volume 227 of *Advances in Intelligent Systems and Computing*, pages 155–169. Springer International Publishing, 2013.

[90] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, Spring 2007.

[91] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima. Reference point specification in hypervolume calculation for fair comparison and efficient search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 585–592. ACM, 2017.

[92] D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., New York, NY, USA, 2001.

[93] H. Kaplan, N. Rubin, M. Sharir, and E. Verbin. Efficient colored orthogonal range counting. *SIAM Journal on Computing*, 38(3):982–1011, June 2008.

[94] K. Klamroth, J. D. Knowles, G. Rudolph, and M. M. Wiecek. Personalized Multiobjective Optimization: An Analytics Perspective (Dagstuhl Seminar 18031). *Dagstuhl Reports*, 8(1):33–99, 2018.

[95] K. Klamroth, R. Lacour, and D. Vanderpooten. On the representation of the search region in multi-objective optimization. *European Journal of Operational Research*, 245(3):767–778, 2015.

[96] J. Knowles and D. Corne. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, volume 1, pages 98–105, 1999.

[97] J. D. Knowles. *Local-search and hybrid evolutionary algorithms for Pareto optimization.* PhD thesis, Department of Computer Science, University of Reading, 2002.

[98] J. D. Knowles, D. W. Corne, and M. Fleischer. Bounded archiving using the Lebesgue measure. In *The 2003 Congress on Evolutionary Computation (CEC'03)*, volume 4, pages 2490–2497, Dec. 2003.

[99] T. Kuhn. *Representative Systems and Decision Support for Multicriteria Optimization Problems.* PhD thesis, Technische Universität Kaiserslautern, 2015.

[100] T. Kuhn, C. M. Fonseca, L. Paquete, S. Ruzika, M. M. Duarte, and J. R. Figueira. Hypervolume subset selection in two dimensions: Formulations and algorithms. *Evolutionary Computation*, 24(3):411–425, Fall 2016.

[101] T. Kuo and S.-Y. Hwang. A genetic algorithm with disruptive selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(2):299–307, Apr. 1996.

[102] R. Lacour, K. Klamroth, and C. M. Fonseca. A box decomposition algorithm to compute the hypervolume indicator. *Computers & Operations Research*, 79:347–360, Mar. 2017.

[103] J. Laitila and A. Moilanen. New performance guarantees for the greedy maximization of submodular set functions. *Optimization Letters*, 11(4):655–665, Apr. 2017.

[104] A. Liefooghe, L. Paquete, and J. R. Figueira. On local search for bi-objective knapsack problems. *Evolutionary Computation*, 21(1):179–196, Spring 2013.

[105] E. M. Lopez, L. M. Antonio, and C. A. Coello Coello. A GPU-based algorithm for a faster hypervolume contribution computation. In A. Gaspar-Cunha et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 9019 of *Lecture Notes in Computer Science*, pages 80–94. Springer International Publishing, 2015.

[106] M. López-Ibáñez, J. Knowles, and M. Laumanns. On sequential online archiving of objective vectors. In R. H. C. Takahashi et al., editors, *International Conference on Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 46–60. Springer Berlin Heidelberg, 2011.

[107] L. Lovász. Submodular functions and convexity. In A. Bachem et al., editors, *Mathematical Programming The State of the Art*, pages 235–257. Springer Berlin Heidelberg, 1983.

[108] H. Markowitz. Portfolio selection. *Journal of Finances*, 7:77–91, 1952.

[109] K. Miettinen. *Nonlinear Multiobjective Optimization.* International Series in Operations Research & Management Science. Springer US, 1998.

[110] K. Miettinen, F. Ruiz, and A. P. Wierzbicki. Multiobjective optimization. In J. Branke et al., editors, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*, chapter Introduction to Multiobjective Optimization: Interactive Approaches, pages 27–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[111] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14(1):265–294, 1978.

[112] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer New York, second edition, 2006.

[113] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag New York, Inc., 1985.

[114] I. Rechenberg. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. *Frommann-Holzboog Verlag, Stuttgart*, 1973.

[115] B. Roy. *Paradigms and Challenges*, pages 19–39. Springer New York, New York, NY, 2016.

[116] G. Rudolph, O. Schütze, C. Grimme, C. Domínguez-Medina, and H. Trautmann. Optimal averaged Hausdorff archives for bi-objective problems: theoretical and numerical results. *Computational Optimization and Applications*, 64(2):589–618, 2016.

[117] G. Rudolph, O. Schütze, and H. Trautmann. On the closest averaged Hausdorff archive for a circularly convex Pareto front. In G. Squillero and P. Burelli, editors, *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016*, volume 9598 of *Lecture Notes in Computer Science*, pages 42–55. Springer International Publishing, 2016.

[118] L. Russo and A. Francisco. Quick hypervolume. *IEEE Transactions on Evolutionary Computation*, 18(4):481–502, Aug. 2014.

[119] L. M. S. Russo and A. P. Francisco. Extending quick hypervolume. *Journal of Heuristics*, 22(3):245–271, June 2016.

[120] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, 1985.

[121] H.-P. Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, 1981.

[122] P. K. Shukla, N. Doll, and H. Schmeck. A theoretical analysis of volume based Pareto front approximations. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO ’14, pages 1415–1422. ACM, 2014.

[123] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, and H. de Garis. An overview of evolutionary computation. In P. B. Brazdil, editor, *Machine Learning: ECML-93*, volume 667 of *Lecture Notes in Computer Science*, pages 442–459. Springer Berlin Heidelberg, 1993.

[124] H. Trautmann, T. Wagner, and D. Brockhoff. R2-EMOA: Focused multiobjective search using R2-indicator-based selection. In G. Nicosia and P. Pardalos, editors, *International Conference on Learning and Intelligent Optimization*, volume 7997 of *Lecture Notes in Computer Science*, pages 70–74. Springer Berlin Heidelberg, 2013.

[125] T. Ulrich, J. Bader, and E. Zitzler. Integrating decision space diversity into hypervolume-based multiobjective search. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO ’10, pages 455–462. ACM, 2010.

[126] T. Ulrich and L. Thiele. Bounding the effectiveness of hypervolume-based $(\mu+\lambda)$-archiving algorithms. In *Learning and Intelligent Optimization*, volume 7219 of *Lecture Notes in Computer Science*, pages 235–249. Springer Berlin Heidelberg, 2012.

[127] L. While. A new analysis of the lebmeasure algorithm for calculating hypervolume. In C. A. Coello Coello et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 326–340. Springer Berlin Heidelberg, 2005.

[128] L. While and L. Bradstreet. Applying the WFG algorithm to calculate incremental hypervolumes. In *CEC 2012, IEEE Congress on Evolutionary Computation*, pages 1–8, June 2012.

[129] L. While, L. Bradstreet, and L. Barone. A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95, Feb. 2012.

[130] L. While, L. Bradstreet, L. Barone, and P. Hingston. Heuristics for optimizing the calculation of hypervolume for multi-objective optimization problems. In *CEC 2005, IEEE Congress on Evolutionary Computation*, volume 3, pages 2225–2232. IEEE, Sept. 2005.

[131] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, Feb. 2006.

[132] K. Yang, A. Emmerich, Michael Deutz, and C. M. Fonseca. Computing 3-D expected hypervolume improvement and related integrals in asymptotically optimal time. In H. Trautmann et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 10173 of *Lecture Notes in Computer Science*, pages 685–700. Springer International Publishing, 2017.

[133] I. Yevseyeva, A. P. Guerreiro, M. T. M. Emmerich, and C. M. Fonseca. A portfolio optimization approach to selection in multiobjective evolutionary algorithms. In T. Bartz-Beielstein et al., editors, *Parallel Problem Solving from Nature — PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 672–681. Springer International Publishing, 2014.

[134] H. Yıldız and S. Suri. On Klee's measure problem on grounded boxes. In *Proceedings of the Twenty-eighth Annual Symposium on Computational Geometry*, SoCG '12, pages 111–120. ACM, June 2012.

[135] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec. 2007.

[136] X. Zhou, N. Mao, W. Li, and C. Sun. A fast algorithm for computing the contribution of a point to the hypervolume. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 4, pages 415–420, Aug. 2007.

[137] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland, 1999.

[138] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In S. Obayashi et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 862–876. Springer Berlin Heidelberg, 2007.

[139] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[140] E. Zitzler, J. Knowles, and L. Thiele. *Quality Assessment of Pareto Set Approximations*, volume 5252 of *Lecture Notes in Computer Science*, chapter 14, pages 373–404. Springer Berlin Heidelberg", 2008.

[141] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In X. Yao et al., editors, *Parallel Problem Solving from Nature — PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer Berlin Heidelberg, 2004.

[142] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In X. Gandibleux et al., editors, *Metaheuristics for Multiobjective Optimisation*, volume 535, pages 3–37. Springer Berlin Heidelberg, 2004.

[143] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. Technical report, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001.

[144] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms – A comparative case study. In A. E. Eiben et al., editors, *Parallel Problem Solving from Nature — PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer Berlin Heidelberg, 1998.

[145] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov. 1999.

[146] E. Zitzler, L. Thiele, and J. Bader. Spam: Set preference algorithm for multiobjective optimization. In G. Rudolph et al., editors, *Parallel Problem Solving from Nature — PPSN X*, pages 847–858. Springer Berlin Heidelberg, 2008.

[147] E. Zitzler, L. Thiele, and J. Bader. On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14(1):58–79, Feb. 2010.

[148] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, Apr. 2003.