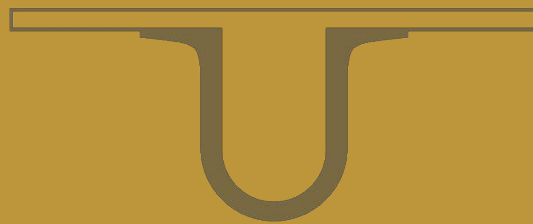




UNIVERSIDADE D
COIMBRA



Maryam Abbasi

MULTIOBJECTIVE SEQUENCE ALIGNMENT
FORMULATION, ALGORITHMS AND APPLICATION

Tese no âmbito do Programa de Doutoramento em Ciências e Tecnologias da Informação, orientada pelo Professor Luís Filipe dos Santos Coelho Paquete, e apresentada no Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Agosto de 2018

Multiobjective Sequence Alignment

Formulation, Algorithms and Application

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

by

Maryam Abbasi

maryam@dei.uc.pt

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

August 2018

Financial support by

Fundação para a Ciência e a Tecnologia

Ref.: SFRH/BD/91451/2012 Multiobjective Sequence Alignment

©2018 Maryam Abbasi

Cover image: © MIKI Yoshihito

(<https://www.flickr.com/photos/mujitra/2559447601/>)



Advisor

Prof. Luís Filipe dos Santos Coelho Paquete

Assistant Professor

Department of Informatics Engineering

Faculty of Sciences and Technology of the University of Coimbra

Dedicated to my family

Abstract

Sequence alignment is a standard technique in bioinformatics to measure the relationship between evolutionary or structurally related DNA/proteins sequences. Most modern programs for sequence alignment optimize a given objective function that is a convex combination of how many gaps need to be inserted into the sequences and how many characters become aligned. Clearly, depending on the weights given to each of the two components, different optimal alignments can be obtained. Therefore, choosing only one weight setting may provide an undesirable bias in further steps of the analysis, such as phylogenetic tree construction, and provide too simplistic interpretations.

In this thesis, we take a different point of view on the mathematical formulation of the *sequence alignment problem*. Rather than considering the optimization of a scalar score function, resulting from a weighted sum of components, we consider a vector score function with the goal of optimizing, *simultaneously*, the different score components. This brings us to the topic of *multiobjective optimization*, which deals with the mathematical formulation of optimization problems with several conflicting objectives as well as with algorithms to solve them. Under this new formulation, these algorithms return a set of non-dominated alignments, each of which representing a trade-off between the several components. This set gives further information about the similarity of the sequences, from which a practitioner could analyze and choose the most plausible alignment.

We consider the biobjective pairwise sequence alignment problem and propose extensions of efficient dynamic programming algorithms for several variants of this problem. We propose a novel pruning technique that substantially reduces the computation time and memory usage. Moreover, we consider a biobjective variant of this problem with more than two sequences, which is computationally intractable. We introduce local search techniques for this problem and conduct an in-depth experimental analysis on a wide range of benchmark instances. Based on the hypervolume indicator and empirical attainment function method-

ology, we establish functional relationships between algorithm performance and instance features. Finally, we present a method that uses multiobjective concepts for the construction of phylogenetic trees. We test this method on two real-life cases and show that the number of distinct phylogenetic tree topologies obtained is very small.

This work shows that multiobjective concepts can successfully be applied to the sequence alignment problem and identifies which approaches can be used for the several variants of this problem. We believe that the methods proposed in this thesis, by providing more information about the relationship between biological sequences than the current known procedures, can be of great value to a broad range of research communities as well as to practitioners in the field.

Keywords: Sequence Alignment, Multiobjective Optimization, Dynamic Programming, Phylogenetic Trees, Combinatorial Optimization.

Resumo

O alinhamento de sequências é um procedimento utilizado na Bioinformática que tem por objetivo medir a semelhança entre sequências de DNA ou proteínas relacionadas entre si de uma forma evolutiva ou estrutural. As aplicações atuais para alinhamento de sequências otimizam uma determinada função objetivo que resulta da combinação convexa da quantidade de espaços a inserir nas sequências e da quantidade de caracteres que ficam alinhados. Dependendo das ponderações atribuídas a cada um destes dois componentes, diferentes alinhamentos podem ser obtidos. Desta forma, a escolha de uma só ponderação pode enviesar, indesejadamente, os passos seguintes da análise, por exemplo, na construção de árvores filogenéticas, e fornecer interpretações demasiado simples.

Esta tese aborda o problema de alinhamento de sequências de uma forma diferente na perspectiva de formulação matemática. Em vez da otimização de uma função escalar que resulta de uma soma ponderada das componentes, considera-se uma função vetorial em que se pretende otimizar, simultaneamente, as suas componentes. O estudo destes problemas é abordado em otimização multi-objetivo, que lida com as formulações matemáticas de problemas de otimização com vários objetivos conflituosos entre si e com algoritmos para a sua resolução. Com esta nova formulação, os algoritmos retornam um conjunto de alinhamentos não-dominados, cada um representando um compromisso entre as várias componentes da função objetivo. Este conjunto, ao fornecer mais informação acerca da semelhança entre as sequências em análise, permite, ao profissional, escolher o alinhamento mais plausível.

Neste estudo, considera-se o problema bi-objetivo de alinhamento emparelhado de sequências e variantes deste problema, para os quais propõem-se extensões de algoritmos eficientes baseados em programação dinâmica. Propõe-se igualmente uma variante bi-objetivo deste problema para mais do que duas sequências, que é considerado um problema computacionalmente intratável. Por esta razão, apresentam-se técnicas de procura local para este problema. Estes algoritmos são analisados experimentalmente num conjunto de instân-

cias de referência. Com base no indicador de hipervolume e na metodologia das funções de aproveitamento, estabelecem-se relações funcionais entre o desempenho dos algoritmos e características destas instâncias. Finalmente, apresenta-se um método que utiliza conceitos de otimização multi-objetivo para a construção de árvores filogenéticas. Este método é testado em dois casos reais. Os resultados obtidos indicam que o número de topologias distintas de árvores filogenéticas é bastante pequeno.

Este estudo mostra que os conceitos multi-objetivo podem ser utilizados com sucesso no problema de alinhamento de sequências e permite identificar quais as abordagens que podem ser utilizadas para cada uma das variantes apresentadas. Ao fornecer mais informação acerca da relação entre as sequências biológicas do que os métodos atuais, espera-se que as contribuições desta tese possam de grande valor tanto para a comunidade acadêmica como para os profissionais de Bioinformática.

Palavras-chave: Alinhamento de Sequências, Otimização Multi-objetivo, Programação Dinâmica, Árvores Filogenéticas, Otimização Combinatória.

Acknowledgements

I want to dedicate these lines to give my thanks to all people whom without them this thesis would not be possible.

First, I want to thank my thesis adviser, Professor Luís Paquete, for believing in me and for giving me the opportunity to work with him on this novel approach, and then for being a great help during my work, with his endless patience, useful advice, and suggestions. I am especially grateful to him for encouraging me to think more comprehensive about the problems and delve deeper to find the best possible solutions.

I would like to thank Pedro Matias for helping me with the code implementations and setup of the web-server MOSAL, as well as to Dr. Miguel Pinheiro for his kind advice on the biological application. I would also like to thank Prof. Stefan Ruzika for hosting me at the University of Koblenz-Landau and for being available to discuss topics of multiobjective optimization and Prof. Francisco Pereira for the very useful discussion on local search procedures for optimization. A final thanks to my friends (Andreia Guerreiro, Dr. Nuno Lourenço, and Pedro Correia) who during the rough moments, cheered me up and gave me the strength to go on.

I owe hugely to my dear parents. Their permanent love and confidence have encouraged me to go ahead in my study and career, who gave me the chance to be in this place and who supported me the most through all my life and education. I hope I will provide them with the reason to be proud of me.

Most importantly, I express my gratitude to my dear husband Pedro Martins whom I met nearly at the start point of my research, married and had two beautiful kids. His support, encouragement, patience and steady love were undeniably the bedrock upon which the past five years of my life have been built. His tolerance to my occasional vulgar moods is a witness in itself of his encouraging devotion and love. I dedicate all my efforts to his and my little ones, Nina and Daniel.

Finally, the research that led to this thesis would not have been possible without the funding and support provided by the Fundação para a Ciência e Tecnologia (FCT) under the scholarship SFRH/BD/91451/2012, by the FCT research project MOSAL - Multiobjective sequence alignment (PTDC/EIA-CCO/098674/2008) and Bilateral project DAAD/CRUP "Representation Systems with Quality Guarantees for Multi-Objective Optimization Problems (RepSys)", and by the Center for Informatics and Systems of the University of Coimbra (CISUC).

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Contributions	5
1.3	Thesis Structure	8
2	Sequence Alignment	11
2.1	Introduction	11
2.2	Genomic Data and Sequence Similarity	12
2.3	The Pairwise Sequence Alignment Problem	13
2.3.1	Notation and Definitions	13
2.3.2	Scoring an Alignment	14
2.3.3	Needleman-Wunsch Algorithm for Global Alignment	16
2.3.4	Global Alignment with Gap Penalties	21
2.3.5	Smith-Waterman Algorithm for Local Alignment	23
2.4	Multiple Sequence Alignment Problem	24
2.4.1	Notation and Definitions	24
2.4.2	Algorithms for the MSA Problem	28
2.4.3	Other MSA Score Functions	37
2.4.4	Benchmarks	40
2.5	Summary and Discussion	41
3	Multiobjective Optimization	43
3.1	Introduction	43
3.2	Notation and Definitions	44
3.2.1	Pareto Optimality	45
3.2.2	Lexicographic Optimality	46

3.2.3	Scalarized Optimality	47
3.3	Solution Methods for Multiobjective Optimization	48
3.3.1	Exact Methods	48
3.3.2	Heuristic Methods	51
3.4	Performance Assessment of Heuristic Methods	65
3.4.1	Indicator of Performance	65
3.4.2	Attainment Function	67
3.5	Summary and Discussion	70
4	Multiobjective Pairwise Sequence Alignment	71
4.1	Introduction	71
4.2	Notation and Definitions	72
4.3	Parametric Sequence Alignment	75
4.4	Algorithms for Multiobjective Pairwise Alignment	75
4.4.1	Dynamic Programming for Multiobjective Pairwise Alignment	76
4.4.2	An ϵ -constraint Algorithm for Multiobjective Pairwise Alignment	87
4.5	Experimental Analysis	95
4.5.1	Performance of the Pruning Technique	95
4.5.2	Computational Analysis with Real Data	97
4.5.3	Computational Analysis with Random Data	99
4.6	Summary and Discussion	100
5	Multiobjective Multiple Sequence Alignment	101
5.1	Introduction	101
5.2	Notation and Definitions	103
5.3	Review of Approaches to the mMSA Problem	104
5.4	Algorithms	106
5.4.1	Pareto Local Search (PLS)	106
5.4.2	Iterated Pareto Local Search (IPLS)	110
5.4.3	NSGA-II	111
5.5	Experimental Analysis	114
5.5.1	Performance Analysis of PLS and IPLS	114
5.5.2	Comparison of IPLS with NSGA-II	120
5.5.3	Comparison of IPLS with Clustal Omega and T-Coffee	121
5.6	Summary and Discussion	125

6	An Application to the Construction of Phylogenetic Trees	127
6.1	Introduction	127
6.2	Introduction to Phylogenetic Trees	128
6.3	Construction of the Phylogenetic Tree	131
6.3.1	UPGMA	131
6.3.2	Neighbor Joining	135
6.3.3	Maximum Parsimony	136
6.3.4	Maximum Likelihood	139
6.3.5	Overall Discussion	139
6.4	A Biobjective Method for Constructing Phylogenetic Trees	140
6.5	Analysis with Real Data	141
6.5.1	First Experiment	141
6.5.2	Second Experiment	142
6.6	Summary and Discussion	143
7	Conclusions and Future Work	149
7.1	Summary and Contributions	149
7.2	Future Work and Open Issues	150
A	Experimental Results for the mMSA	153

List of Figures

1.1	A trade off between time and cost	3
2.1	An alignment in a $2 \times \ell$ -matrix	14
2.2	Illustration of the DP matrix of the Needleman-Wunsch algorithm and the traceback of the alignment (red arrows)	20
2.3	A comparison of the global versus local alignment (Jones and Pevzner, 2004, page 182)	25
2.4	All pairwise alignments	31
2.5	The ClustalW guide tree	32
2.6	T-Coffee primary library, with all pairwise alignments and the Primary Weight value (PW)	34
2.7	The three possible alignments of SeqA and SeqB for our example	35
3.1	Mapping of the n -dimensional space to the q -dimensional objective space . .	44
3.2	A schematic illustration of the “dominance relation”, assuming two maximiz- ing objectives	45
3.3	Illustration of the weighted-sum method	49
3.4	A schematic illustration of the ϵ -constraint method, assuming biobjective op- timization, maximizing f_1 subject to the constraint f_2	50
3.5	Illustration of single point (left) and double point (right) crossover operator .	54
3.6	Illustration of mutation operator	54
3.7	Illustration of Pareto ranking in MOGA	57
3.8	Illustration of ranking (left) and crowding distance (right) in NSGA-II . . .	60
3.9	Illustration of the hypervolume indicator (left) and the ϵ -indicator (right) . .	67
3.10	Illustration of the empirical attainment function	68
3.11	Hypothetical outcomes of the three runs for a MOP instance	69

4.1	Various alignments of sequences AAGTAGC and ATGACG (left) and their images in the score space (right)	74
4.2	Illustration of the lower bound set. Lower bound set contains the three pre-computed non-dominated score vectors MIN, MID and MAX. Any alignment whose score vector is inside of the shaded area cannot be Pareto optimal, since it would be dominated by MIN, MID or MAX	81
4.3	The experimental results for DP-Prune for Problems (VSDP) and (VSGP) with random data set ranged from $n = 200$ to 3000	99
5.1	Mutation operator: gaps are randomly chosen and shifted to another position. Columns with only indels are removed	112
5.2	Single point crossover operator: The first parent alignment (X) is cut straight at a randomly chosen position, and two blocks are created (X1 X2). The second parent alignment (Y) is tailored so that the right piece can be joined to the left piece of the first parent (Y1 Y2). Then, the blocks of (Y1 Y2) are filled with indels in order to match the size and keep the order of the letters in the sequences. In the end, the blocks of two parents are crossed (X1 Y2 and Y1 Y2)	113
5.3	EAF difference plot for instance BB11008 from family group RV11.	123
5.4	EAF difference plot for instance BB20003 from family group RV12.	123
6.1	Different representations of a phylogenetic tree.	129
6.2	Illustration of bootstrapping method for a phylogenetic tree.	130
6.3	The distance matrix of three sequences with length 6. The distance is calculated as the number of substitutions between sequences divided by their lengths.	131
6.4	Steps of the UPGMA method	132
6.5	Steps of Neighbor Joining method	134
6.6	Informative and non-informative sites in maximum parsimony method.	137
6.7	Principles of tree construction by the maximum parsimony method. Tree 1 is the most parsimonious tree because its topology is based on the minimum number of substitutions.	137
6.8	Phylogenetic trees for the first experiment: biobjective model for a gap value of 283 (a) and Maximum Likelihood model (b).	144

6.9	Staircase line representation of the non-dominated score sets for the first experiment.	145
6.10	Phylogenetic trees for the second experiment: biobjective model for a gap value of 22 (a) and 54 (b) and Maximum Likelihood model (c).	146
6.11	Staircase line representation of the non-dominated score sets for the second experiment.	147
A.1	Plot of the EAF for the instance BB11001	154
A.2	Plot of the EAF for the instance BB11002	154
A.3	Plot of the EAF for the instance BB11003	155
A.4	Plot of the EAF for the instance BB11004	155
A.5	Plot of the EAF for the instance BB11005	156
A.6	Plot of the EAF for the instance BB11006	156
A.7	Plot of the EAF for the instance BB11007	157
A.8	Plot of the EAF for the instance BB11008	157
A.9	Plot of the EAF for the instance BB11009	158
A.10	Plot of the EAF for the instance BB11010	158
A.11	Plot of the EAF for the instance BB12001	159
A.12	Plot of the EAF for the instance BB12002	159
A.13	Plot of the EAF for the instance BB12003	160
A.14	Plot of the EAF for the instance BB12004	160
A.15	Plot of the EAF for the instance BB12005	161
A.16	Plot of the EAF for the instance BB12006	161
A.17	Plot of the EAF for the instance BB12007	162
A.18	Plot of the EAF for the instance BB12008	162
A.19	Plot of the EAF for the instance BB12009	163
A.20	Plot of the EAF for the instance BB12010	163
A.21	Plot of the EAF for the instance BB20001	164
A.22	Plot of the EAF for the instance BB20002	164
A.23	Plot of the EAF for the instance BB20003	165
A.24	Plot of the EAF for the instance BB20004	165
A.25	Plot of the EAF for the instance BB20005	166
A.26	Plot of the EAF for the instance BB20006	166
A.27	Plot of the EAF for the instance BB20007	167

A.28 Plot of the EAF for the instance BB20008	167
A.29 Plot of the EAF for the instance BB20009	168
A.30 Plot of the EAF for the instance BB20010	168
A.31 Plot of the EAF for the instance BB30001	169
A.32 Plot of the EAF for the instance BB30002	169
A.33 Plot of the EAF for the instance BB30003	170
A.34 Plot of the EAF for the instance BB30004	170
A.35 Plot of the EAF for the instance BB30005	171
A.36 Plot of the EAF for the instance BB30006	171
A.37 Plot of the EAF for the instance BB30007	172
A.38 Plot of the EAF for the instance BB30008	172
A.39 Plot of the EAF for the instance BB30009	173
A.40 Plot of the EAF for the instance BB30010	173
A.41 Plot of the EAF for the instance BB40001	174
A.42 Plot of the EAF for the instance BB40002	174
A.43 Plot of the EAF for the instance BB40003	175
A.44 Plot of the EAF for the instance BB40004	175
A.45 Plot of the EAF for the instance BB40005	176
A.46 Plot of the EAF for the instance BB40006	176
A.47 Plot of the EAF for the instance BB40007	177
A.48 Plot of the EAF for the instance BB40008	177
A.49 Plot of the EAF for the instance BB40009	178
A.50 Plot of the EAF for the instance BB40010	178
A.51 Plot of the EAF for the instance BB50001	179
A.52 Plot of the EAF for the instance BB50002	179
A.53 Plot of the EAF for the instance BB50003	180
A.54 Plot of the EAF for the instance BB50004	180
A.55 Plot of the EAF for the instance BB50005	181
A.56 Plot of the EAF for the instance BB50006	181
A.57 Plot of the EAF for the instance BB50007	182
A.58 Plot of the EAF for the instance BB50008	182
A.59 Plot of the EAF for the instance BB50009	183
A.60 Plot of the EAF for the instance BB50010	183

List of Tables

2.1	An alignment with length 5 and its position weight matrix.	40
4.1	Upper bounds for the example in Section 4.4.1	83
4.2	Matrix \mathbf{P} without (left) and with (right) pruning for the example in Section 4.4.1	84
4.3	Matrices \mathbf{P} and \mathbf{Q} for $k = 0$ and $k = 1$ for the example in Section 4.4.1 with the ϵ -constraint approach. The numbers in the blue boxes show the traceback for the alignment with score $(-3, -1)$	91
4.4	ϵ -constraint matrices for Problem (VSGP) with $k = 1$ and $k = 2$ for the example in Section 4.4.1	94
4.5	Experimental results for the pruning technique for Problem (VSDP)	96
4.6	Experimental results for the pruning technique for Problem (VSGP)	96
4.7	Experimental results on the two data sets	98
5.1	Single objective methods that are used to create starting alignments	107
5.2	Average relative hypervolume indicator value (I_{RHV}) for PLS for several k -block neighborhood sizes and starting solutions (Clustal Omega , T-Coffee and Rand) for each instance of the dataset RV11. The results are averaged over 30 runs. See text for more details	116
5.3	Average relative hypervolume indicator value (I_{RHV}) for PLS with several k -block neighborhood sizes and starting solutions (Clustal Omega , T-Coffee and Rand) for for each instance of the dataset RV20. The results are averaged over 30 runs. See text for more details	117
5.4	Average relative hypervolume indicator value (I_{RHV}) for IPLS with several perturbation iterations (P) for each instance of dataset RV11. The results are averaged over 30 runs. See text for more details	118

5.5	Average relative hypervolume indicator value (I_{RHV}) for IPLS with several perturbation numbers (P) for each instance of the dataset RV20. The results are averaged over 30 runs. See text for more details	119
5.6	Average relative hypervolume indicator value (I_{RHV}) for IPLS and NSGA-II for the first 10 instances of the datasets RV11, RV12, RV20, RV30, RV40 and RV50. The results are averaged over 30 runs for each algorithm. See text for more details	122
5.7	The selective datasets from difference reference of the benchmark.	124
5.8	The results of SP score in T-Coffee, Clustal Omega and IPLS on the selected test cases.	124
5.9	The results of TC score in T-Coffee, Clustal Omega and IPLS on the selected test cases.	124

List of Abbreviations

Abbreviation	Meaning
AF	Attainment Function
CWAC	Component Wise Acceptance Criterion
DP	Dynamic Programming
EA	Evolutionary algorithm
EAF	Empirical Attainment Function
FPS	Fitness Proportionate Selection
HTU	Hypothetical Taxonomic Unit
ILS	Iterated Local Search
IPLS	Iterated Pareto Local Search
LS	Local Search
MEA	Multiobjective Evolutionary Algorithms
ML	Maximum likelihood
mMSA	Multiobjective Multiple Sequence Alignment
MOGA	Multiobjective Genetic Algorithm
MOP	Multiobjective Optimization Problem
mPSA	Multiobjective Pairwise Sequence Alignment
MP	Maximum Parsimony
MSA	Multiple Sequence Alignment
NJ	Neighbor Joining
NSGA	Non-dominated Sorting Genetic Algorithm
NSGA-II	Fast Non-dominated Sorting Genetic Algorithm
OTU	Operational Taxonomic Unit
PAES	Pareto-Archived Evolution Strategy
PLS	Pareto Local Search

Abbreviation Meaning

PSA	Pairwise Sequence Alignment
PW	Primary Weights
SAC	Scalarized Acceptance Criterion
SP-score	Sum-of-pairs score
SPEA	Strength Pareto Evolutionary Algorithm
SUS	Stochastic Universal Sampling
VEGA	Vector Evaluated Genetic Algorithm
WSP-score	Weighted Sum-of-Pairs score

Chapter 1

Introduction

1.1 Motivation

Sequence alignment is the first step in evaluating the degree of similarity between biological sequences, such as DNA or proteins. An alignment allows highlighting the differences between potentially similar biological sequences, which may result from mutations and can be interpreted in evolutionary terms. Based on the information provided by the alignment, it is possible to explain and predict functional and structural information of a sequence. For this reason, finding biologically meaningful alignments is a fundamental aspect in Bioinformatics.

A DNA sequence is made of 4 different repeating units called nucleotides (Egli and Saenger, 2013) and a protein is produced of 20 units called amino acids (Nelson et al., 2008). In biology, a single unit (nucleotide/amino acid) that makes up the DNA or protein is called *residue*. The assessment of the relationship between two or more sequences is performed by a sequence alignment procedure, which consists of aligning the sequences in order to optimize a given score function that takes into account the numbers of identical or similar residues between the sequences.¹

¹Similar residues are those that have similar chemical characteristics. For instance, in proteins, substitution of amino acid residues by chemically equivalent ones often does not have a substantial effect on the structure or on the function of that protein.

There are some rules on how to build the score function of a sequence alignment procedure such that the alignment produced is meaningful in a biological context. For instance, there is a wide consensus that the score function should take into account the matching between identical or similar residues from the different sequences as much as possible. However, matching comes with a cost: very often, blocks of residues must be detached from each other in order to maximize the matching. Depending on how the matching and the number of detached blocks of residues are handled in the sequence alignment procedure, different alignments may be obtained.

The following example shows a possible alignment between the sequences GCPVSSPNVEM and GCPYGCPEMDA.

```

1:  GCPVS-SPNVEM
    |||**-*|****
2:  GCPYGCPEMDA

```

The identical residues (*matches*) between the two sequences are marked by the character ‘|’ and the unequal ones (*mismatches/substitutions*) are shown with character ‘*’. Note that one of the residues in the second sequence does not have a corresponding residue in the first, which is marked by the character ‘-’ in the alignment (an *indel*). Another possible alignment of the same sequences is shown below.

```

1:  GCPVSS-PNVEM--
    |||***-|--||--
2:  GCPYGCDP--EMDA

```

The quality of the alignment is assessed by means of a score function, which can be as simple as subtracting the number of indels from the number of matches in the alignment. In the two examples above, this score function would return $4 - 3 = 1$ for the first alignment and $5 - 5 = 0$ for the second alignment. Note that the second alignment has more matches than that of the first, but this is achieved by inserting more indels. When introducing indels in the alignment, the following question may arise: How many indels can be introduced? Actually, by introducing a large number of indels, the similarity would increase, but would that be biologically relevant? (See discussion in (Morrison, 2015))

In general, the score function in sequence alignment is defined as a combination of different components with different weights. Each component, such as the number of matches,

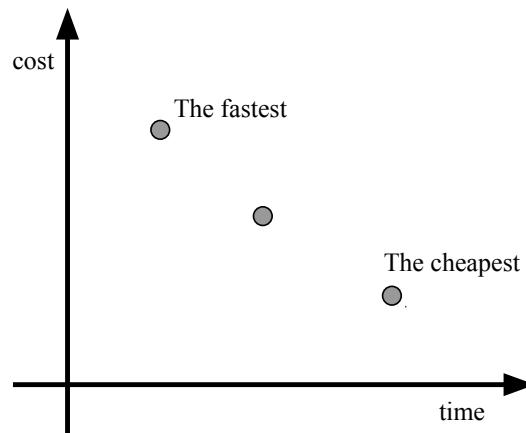


Figure 1.1: A trade off between time and cost

mismatches, and indels can be assigned a positive weight. For instance, giving more weight to the number of matches, the score function would favor alignments with very large similarity, even if a large number of indels is required. If more weight is given to the number of indels, then the score function would promote alignments with very few numbers of indels, which may result in low similarity. Clearly, the choice of different weights may produce distinct alignment, but, unfortunately, there is no explicit agreement on how to choose those weights. Therefore, an alignment generated by a computer program such as Blast or FASTA represents only one of many possibilities that can be obtained in optimizing the given score function.

In this thesis, we take a different point of view on the mathematical formulation of the *sequence alignment problem*. Rather than considering a scalar score function, resulting from a weighted sum of components, we consider a vector score function with the goal of optimizing, *simultaneously*, the different score components. This brings us to the topic of *multiobjective optimization*, which deals with the mathematical formulation of optimization problems with several conflicting objectives as well as with algorithms to solve them.

Multiobjective optimization problems arise naturally in many real-world optimization scenarios, for example, when planning a trip from a city X to another city Y . There may be different paths between two cities with different objectives: distance, cost, travel time, etc. A solution to this problem is the one which is optimal with respect to all objectives. But is there such a *ideal* solution? The cheapest path is not necessarily the fastest, nor the fastest

needs to be the cheapest if tolled highways are considered.

When the decision maker is cannot express his preferences *a priori*, the notion of *Pareto* optimality is the most natural to be used when dealing with multiobjective optimization. In this case, one is interested in obtaining a set of solutions that represent the optimal trade-off between the different objectives, i.e., solutions which are not worse than any other and strictly better in at least one of the objectives. Figure 1.1 shows a set of optimal solutions to the problem of finding a path between two cities that minimize both cost and time. The leftmost point corresponds to the fastest solution whereas the rightmost point corresponds to the cheapest solution; the point in the middle corresponds to a solution that is as relevant as the other two since it is not so fast as the cheapest and not so expensive as the fastest. Finding this optimal trade-off between the objectives is commonly known as solving the multiobjective problem in terms of Pareto optimality.

From our point of view, the sequence alignment problem is a natural multiobjective optimization problem – it deals with the conflicting goal of maximizing similarity and minimizing the insertion of indels. A multiobjective formulation of the sequence algorithm problem has the advantage of requiring no setup of weights. However, there might be several *incomparable* alignments that have to be considered in the analysis, but we consider this to be a natural consequence of the way the problem is considered in practice. The applications of these concepts to sequence alignment were mentioned, for the first time, in Roytberg et al. (1999). Up to our knowledge, no other published work explored the same concepts until the starting of this thesis, in 2011.

Finding such a set of optimal solutions to multiobjective optimization problems is a big challenge in computer science. Many such problems are known to be NP-hard even if their single objective versions can be solved in a polynomial amount of time. However, for the multiobjective formulation of the sequence alignment problem, it is known that this is not the case if only two sequences are considered in Roytberg et al. (1999). This is an appealing feature not only for practitioners but also for the scientific community in multiobjective optimization.

In the following, we introduce the main research questions that motivate this work.

Q1 - In Roytberg et al. (1999), a simple dynamic programming approach is proposed to solve the multiobjective optimization problem of maximizing matches and minimizing indels with two sequences. However, in real-life applications, *gaps* are usually considered instead of indels. A gap is a consecutive set of indels and may have a score value that

is not a linear function of its length. In addition, the score functions that are used in computer programs for sequence alignment rely on *substitution score matrices* to determine the score of aligning two distinct characters. How to adapt the dynamic programming approach to these new problem formulations?

- Q2** - The dynamic programming approach proposed in [Roytberg et al. \(1999\)](#) solves a sequence of subproblems that takes a polynomial amount of time. However, some of the subproblems do not need to be solved since their solution do not probably lead to optimal alignments. Can pruning conditions be determined for this problem, for instance, based on upper and lower bounds of partial alignments? This is a standard procedure in branch and bound.
- Q3** - A generic way of solving multiobjective optimization problems is to solve one objective while the remaining are used as constraints. This is known as ϵ -constraint method. Can this paradigm be also used for solving the multiobjective sequence alignment problem?
- Q4** - When more than two sequence alignments are considered, the problem becomes NP-hard, which means that we do not expect to solve the problem in a polynomial amount of time. For this reason, it is natural to consider heuristic approaches. A known approach to solve multiobjective optimization problems is called Pareto Local Search ([Paquete et al., 2007](#)), which is based on the local search paradigm. How to adapt this framework to solve this problem?
- Q5** - Sequence alignment is commonly used to construct a phylogenetic tree, which gives further insight into the evolutionary relationships among several species. It is expected that multiobjective sequence alignment gives rise to several phylogenetic trees. How to generate such trees and how to analyze them?

1.2 Contributions

This thesis aims at solving the research challenges described in the previous section. The contributions are mostly of algorithmic and methodological nature. In summary, the main contributions of this thesis are as follows.

- C1** - We propose extensions of the dynamic programming approach proposed in [Roytberg et al. \(1999\)](#) for different multiobjective problem formulations with two sequences, for instance, considering gaps and substitution matrices.

- C2** - We define pruning techniques that considerably reduce the number of subproblems to be solved on the dynamic programming approach. Our experimental results show that the pruning technique can improve the computational time up to 80% and memory storage up to 90% on well-known benchmark data sets.
- C3** - We present an ϵ -constraint technique to solve the multiobjective sequence alignment problem for two sequences. The proposed algorithm maximizes the substitution score with an equality constraint on the number of indels or gaps.
- C4** - We propose several heuristic approaches, based on local search procedures, to solve the problem when more than two sequences are considered. Several algorithm design options are discussed and analyzed, with particular emphasis on the starting alignment and neighborhood search. A novel perturbation technique is proposed to improve the local search. In addition, for comparison purpose, we have adapted NSGA-II technique (Deb et al., 2002), a well-known multiobjective evolutionary algorithm, to this problem.
- C5** - We propose a new method for the construction of phylogenetic trees based on the multiobjective sequence alignment. We illustrate the application of this method on known sequence data to show the effectiveness of this approach in practice. Noteworthy, the number of phylogenetic trees is shown to be very low, and the analysis gave a further interesting insight into the relations between species.

In this work, we prove the correctness of our algorithms, discuss their time complexity and analyze their performance in practice on well-known benchmark data sets. The most effective dynamic programming algorithms presented in this thesis for the sequence alignment problem with two sequences are publicly available at the website <http://mosal.dei.uc.pt>, which also allows the user to run them online and to visualize the set of alignments.

Most of the contributions were published in international journals and presented at international conferences. They are listed next, in chronological order for each type of venue, together with reference to the contribution.

Journals

- P1** - M.Abbasi, L.Paquete, A.Liefooghe, M.Pinheiro, P.Matias, *Improvements on bicriteria pairwise sequence alignment: algorithms and applications*. Bioinformatics, 29(8):996-1003, 2013 (Abbasi et al., 2013a). See contributions C1, C2 and C5.

- P2** - L.Paquete, P.Matias, M.Abbasi, M.Pinheiro, *MOSAL: Software tools for multiobjective sequence alignment*. Source Code for Biology and Medicine, 9(2), 2014 (Paquete et al., 2014). See contributions C1 and C2.
- P3** - M.Abbasi, L.Paquete, F.Pereira, *Heuristics for multiobjective multiple sequence alignment*. BioMedical Engineering Online, 15(70), 2016 (Abbasi et al., 2016). See contribution C4.

Conferences

- P4** - M.Abbasi, L.Paquete, A.Liefooghe, M.C.Dias, *Multiobjective sequence alignment: Formulation and Algorithms*. The 19th International Conference on Intelligent Systems for Molecular Biology and the 10th European Conference on Computational Biology (ISMB ECCB 2011), 2011 (Abbasi et al., 2011) (with poster). See contributions C1 and C2.
- P5** - M.Abbasi, L.Paquete, M.Pinheiro, *Dynamic programming algorithms for biobjective sequence alignment*. The 1st Conference on the Bioinformatics and Computational Biology, Bioinformatics Open Days (BOD 2012), 40, 2012 (Abbasi et al., 2012) (with poster). See contributions C1 and C2.
- P6** - L.Paquete, M.Abbasi, M.Pinheiro, P.Matias, *Algorithms for multiobjective sequence alignment*. The 2nd Workshop on Bio-Optimization, 2012 (Paquete et al., 2012b) (with oral communication). See contributions C1 and C2.
- P7** - L.Paquete, M.Abbasi, F.Pereira, P.Matias, *Algorithms and applications of biobjective pairwise sequence alignment*. The 2012 Mini EURO Conference on Computational Biology, Bioinformatics and Medicine (EURO-CBBM 2012), 2012 (Paquete et al., 2012a) (with oral communication) . See contributions C1 and C2.
- P8** - M.Abbasi, L.Paquete, F.Pereira, S.Schenker, *Local search for bicriteria multiple sequence alignment*. The 5th German Conference on Bioinformatics, 102, 2013 (Abbasi et al., 2013b) (with poster). See contribution C4.
- P9** - M.Abbasi, L.Paquete, F.Pereira, *Local search for multiobjective multiple sequence alignment*. The 3rd Work-Conference on Bioinformatics and Biomedical Engineering (IWB-BIO 2015), Lecture Notes in Computer Science Vol. 9044, 175-182, Springer, 2015 (Abbasi et al., 2015) (with oral communication). See contribution C4.

During this work, I also gave several talks about the main topic of this thesis.

- *On multicriteria sequence alignment*, Department of Informatics Engineering, University of Coimbra, April 2011.
- *On multiobjective sequence alignment: formulation and algorithms*, Mathematical Institute, University of Koblenz-Landau, Germany, November 2012.
- *Algorithms and applications of biobjective pairwise sequence alignment*, Department of Mathematics, University of Coimbra, December 2012.
- *Advances in multiobjective sequence alignment*, Department of Informatics Engineering, University of Coimbra, June 2013.

In addition, I participated in the following summer schools.

- Artificial Evolution Summer School (AESS2013), June 2013, Quiberon, France
- Summer School in Computational Biology, September 2016, Coimbra, Portugal

From 2011 to 2013, I was awarded a Ph.D. grant within the project *MOSAL - Multiobjective sequence alignment*, a research project funded by FCT (PTDC/EIA-CCO/098674/2008). In addition, I was a team member of the project *RepSys - Representation systems with quality guarantees for multiobjective optimization problems*, a Germany/Portugal Bilateral Cooperation Research Project funded by DAAD/CRUP, from 2012 to 2013. In the context of this project, I had the opportunity of visiting Prof. Dr. Stefan Ruzika and his group at Mathematical Institute from the University of Koblenz-Landau, Germany. Finally, from 2013 to 2017, I was granted and supported by the scholarship SFRH/BD/91451/2012 from FCT, which without it this thesis would not be possible.

1.3 Thesis Structure

This thesis is divided into three main parts: the first is concerned with introductory notions and a review of the literature associated with sequence alignment and multiobjective optimization problems and algorithms (Chapter 2 and 3). The second covers the algorithmic contributions of this thesis, in particular for pairwise sequence alignment (Chapter 4) and multiple sequence alignment (Chapter 5). The third part discusses the real-life application of multiobjective concepts for constructing phylogenetic trees. In the following, we provide an outline of this thesis, together with a brief description of the main contributions.

- *Chapter 2 - Sequence Alignment.*

This chapter introduces the main concepts of sequence alignment and how to evaluate, or score, its quality. It also covers the different methods for solving the several variants of this problem. What is a biological sequence? What is a sequence alignment? How to score an alignment? What is a pairwise and a multiple sequence alignment? These questions are answered in Chapter 2 which introduces the different methods of scoring and explains proposed single objective methods to solve them.

- *Chapter 3 - Multiobjective Optimization.*

This chapter covers the topic of multiobjective optimization. It introduces notations and definitions required to define different notions of optimality under several conflicting objectives, as well as solution methods, exact and heuristic, to solve this problem. In this chapter, the notations, definitions, solutions methods (exact and heuristic) as well as performance assessment in order to compare the different heuristic methods are explained in details.

- *Chapter 4 - Multiobjective Pairwise Sequence Alignment.*

Chapter 4 introduces the multiobjective formulation for pairwise sequence alignment. It describes dynamic programming approaches for the several variants and a pruning technique to improve computation time and space. Thereafter, we introduced an approach based on the epsilon-constraint paradigm. This chapter contains the contributions of C1 and C2.

- *Chapter 5 - Multiobjective Multiple Sequence Alignment.*

Chapter 5 presents the multiobjective formulation of the sequence alignment problem when more than two sequences are considered. A local search method and an adaptation of NSGA-II are proposed to tackle this problem. A thorough experimental analysis is performed in order to compare the two heuristic methods. This chapter contains the contributions of C3 and C4.

- *Chapter 6 - An Application to the Construction of Phylogenetic Trees*

In this chapter, we turn our attention to the use of multiobjective pairwise sequence alignment to construct phylogenetic trees. In particular, it proposes a new method to construct the tree based on the optimal alignments obtained from pairs of sequences. This chapter contains the contributions of C5.

- *Chapter 7 - Conclusions and Future Work*

Chapter 7 presents the final conclusions of this thesis and discusses further research directions of this work.

Chapter 2

Sequence Alignment

2.1 Introduction

Sequence alignment is one of the most frequent tasks in Bioinformatics is (Notredame, 2002). Procedures that rely on sequence comparison are diverse and range from database searches to secondary structure prediction (Jones, 1999). Sequences can be compared in pairs to find gene function, or multiple of them can be aligned in order to visualize the effect of evolution across a whole protein family (Notredame, 2002).

In this chapter, we start by introducing the problem of finding sequence similarity between the two sequences, the so-called *pairwise alignment problem*. Then, we describe exact algorithms based on the dynamic programming paradigm to solve this problem. In addition, we present extensions of this algorithm to solve variants of this problem: global alignment, for comparing the complete sequences, local alignment, for comparing only some parts of the sequences, and alignment with gap penalties. At last, we introduce the problem of finding sequence similarity for more than two sequences, the *multiple sequence alignment*, which is a much more challenging problem from a computational point of view. We describe an exact algorithm based on dynamic programming as well as some well-known heuristic approaches.

2.2 Genomic Data and Sequence Similarity

DNA is understood as a sequence, or string, that contains characters from the alphabet {A, C, G, T} (known as *nucleotides*). Similarly, proteins can be seen as sequences taken from an alphabet of 20 characters (known as *amino acids*). If two DNA or amino acid arrangements are similar, there is a chance that they are *homologous*, that is, they share common evolutionary roots. Comparing homologous sequences is a fundamental step in Bioinformatics to detect functional regions (La et al., 2005) and to reconstruct evolutionary histories (Notredame, 2002; Durbin et al., 1998). The relationships between sequences are very complex since they have been exposed to evolutionary mutations over millions of years. Nevertheless, the first step towards understanding this is to look for *sequence similarity*. To detect if the two sequences are similar, one has to align them properly.

When sequences develop from a common ancestor, their characters (residues¹) can go through *substitutions* (characters are replaced by some other characters), *insertions* (new characters added to a sequence besides to the existing ones) and *deletions* (some characters disappear) (Isaev, 2004). Sometimes, the insertion or deletion (*indels*) occurs in an entire subsequence sequence as a single mutational event. Many of these single mutational events can have quite different sizes (Gusfield et al., 1994). Therefore, when aligning two sequences, characters must be allowed to be aligned not only with other characters but also with indels, which we denote with the ‘-’ character. However, by considering indels, there may be many more possible alignments as shown in the following example:

Example 2.1. *An example of alignments between two sequences TACCAGT and CCCGTAA:*

TACCAGT	TACCAGT--	TACCAGT--
CCCGTAA	C-CC-GTAA	--CCCGTAA

In Bioinformatics, sequence similarity has been used in different applications:

1. *Finding homology.* One of the core purposes of sequence alignment is to find *homology*. Homology means that two sequences share a mutual ancestor. Discovering homology amongst organisms may enable us to use information of one known sequence to other sequences, or to infer the purpose of one organism’s gene from that of a related species.

¹ A residue is a single unit within a polymer, such as an amino acid inside a polypeptide or protein.

2. *Defining the origin of a sequence.* If a DNA or protein sequence is found, but its originating species is undetermined, sequence alignment can be used to detect likely sources, that is, the known sequences that most closely match the sequence.
3. *Constructing Evolutionary Trees.* From homology data, evolutionary (phylogenetic) trees can be built. A phylogenetic tree is a branching diagram (tree) that shows the inferred evolutionary relationships among various biological species upon similarities and differences in their physical or genetic characteristics.

2.3 The Pairwise Sequence Alignment Problem

In this section, we introduce the problem of calculating an alignment between two sequences. This problem is known in the literature as *Pairwise Sequence Alignment* (PSA). We describe a classical dynamic programming algorithm for this problem. Then, we explain the generalization of this method for the comparison of two sub-sequences. Comparing the similarity of the entire sequences is called *global alignment*, whereas for similar sub-sequences is known as *local alignment*.

2.3.1 Notation and Definitions

To establish the degree of similarity, the sequences need to be aligned. However, how to choose between several possible alignments? To answer the question, we need to find a way to score all possible alignments. In order to mathematically define an alignment and define a measure of similarity between sequences, some formal definitions need to be introduced. A formal definition of an alignment is given as follows (Bockenhauer and Bongartz, 2007):

Definition 2.3.1. *Pairwise sequence alignment.*

Let $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$ be two strings over an alphabet Σ . Let $'-'$ $\notin \Sigma$ be an indel character, $\Sigma' = \Sigma \cup \{'-\'}$ and λ denote an empty string. Let $h : (\Sigma')^* \rightarrow \Sigma^*$ be a homomorphism defined by $h(a) = a$ for all $a \in \Sigma$, and $h('-') = \lambda$. An alignment φ of A and B is a pair (A', B') of strings of length $\ell \geq \max\{n_1, n_2\}$ over the alphabet Σ' , such that the following conditions hold:

1. $h(A') = A$
2. $h(B') = B$

	1	2	3	4	5	6	7	8	9
T	A	C	C	A	G	T	-	-	
C	-	C	C	-	G	T	A	A	

Figure 2.1: An alignment in a $2 \times \ell$ -matrix

3. there is no position containing an indel character in both A' and B' , i.e., if a'_i (b'_i) indicates the i -th position in A' (B') then $a'_i \neq '-'$ and $b'_i \neq '-'$ for all $i \in \{1, \dots, \ell\}$.

Conditions 1 and 2 in the definition above state that deleting all indel characters from A' and B' yields the string A and B , respectively. Condition 3 states that it is not possible to have two indels in the same position of the alignment.

For the sake of explanation, we represent a pairwise alignment φ in a $2 \times \ell$ -matrix in such a way that the sequences of the alignment are written one below the other. Also, an alignment can be represented as a sequence of ordered pairs, $\varphi = (\varphi_1, \dots, \varphi_\ell)$; we use these representations interchangeably, depending on the context. Figure 2.1 shows the second alignment of Example 2.1 (in Page 12) in a matrix format. In an alignment each column can be classified as follows:

- *Match*: Both characters are the same (columns 3, 4, 6 and 7 in Figure 2.1).
- *Mismatch*: Both characters are not the same and there is no indel '-' character in both (column 1 in Figure 2.1).
- *Substitution*: There is no indel '-' character in both (column 1, 3, 4, 6 and 7 in Figure 2.1).
- *Indel*: Either the character in the first row or in the second row is an indel (columns 2, 5, 8 and 9 in Figure 2.1).

Note that there is a *gap* when consecutive columns of indels occur (columns 8 and 9 from a gap in Figure 2.1). Gaps are represented as contiguous indels in a sequence alignment.

2.3.2 Scoring an Alignment

To evaluate the quality of an alignment, there is a need to define a score function. The most popular score function assumes independence among the columns, and the total score

of the alignment is set to be equal to the sum of the scores of each column. Therefore, for such functions, only the scores for indel, match/mismatch or substitution need to be defined (Gusfield, 1999; Bockenhauer and Bongartz, 2007).

To score a given alignment φ between two strings, at least two parameters are required:

- A score for each indel in the alignment matrix, i.e., for each column in the alignment matrix that contains an indel ‘-’ character.
- A weight for each column in the alignment matrix that does not contain an indel.

In the latter case, a *substitution matrix* is used, which gives a score for exchanging one character in a sequence with another character in the other sequence. For amino acids, a substitution matrix assigns scores or frequencies to the alignment of each possible pair of amino acids, usually based on the similarity of the chemical properties the amino acids’ and/or the evolutionary probability of the mutation. The substitution matrices for amino acids are available as standard 20×20 matrices. The two most well-known matrices are PAM and BLOSUM families (Isaev, 2004). For nucleotides sequences, typically a much simpler substitution matrix of size 4×4 is used, which contains two weights: matches (α) and mismatches (β).

The score of an alignment can be simply be defined as follows (Bockenhauer and Bongartz, 2007):

Definition 2.3.2. *Score of a pairwise alignment.*

Let $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$ be two sequences over an alphabet Σ . Let $\varphi = (\varphi_1, \dots, \varphi_\ell)$ be an alignment with length ℓ . For $a, b \in \Sigma$, let $s(\varphi_j)$ indicate the score of the j -th column (i.e., a pair such as $((a, '-'), ('-', b))$ or (a, b)) in the alignment, $1 \leq j \leq \ell$. Let M be a substitution matrix for aligning two characters and let θ be the score for a pair that contains $(a, '-')$ or $(('-', b))$. The score function $\delta(\varphi)$ of the alignment φ is given by:

$$\delta(\varphi) = \sum_{j=1}^{\ell} s(\varphi_j) \quad (2.1)$$

where,

$$s(\varphi_j) = \begin{cases} M[a, b] & \text{if } \varphi_j = (a, b), \\ \theta & \text{otherwise} \end{cases} \quad (2.2)$$

A very common particular case of scoring an alignment φ is to count the number of indels ($d(\varphi)$), matches ($m(\varphi)$) and mismatches ($ms(\varphi)$). Different weights can be defined to express the relative importance of an indel, match or mismatch. Therefore, a particular case of the score function given above is given by the following expression:

$$\delta(\varphi) = \alpha \cdot m(\varphi) + \beta \cdot ms(\varphi) + \theta \cdot d(\varphi) \quad (2.3)$$

Usually, β and θ are non-positive real numbers, whereas α is a non-negative real number. Consider the second alignment of Example 2.1 in Section 2.2 (page 12). Let $m(\varphi)$, $ms(\varphi)$ and $d(\varphi)$ be equal to 4, 1 and 4 respectively. Let α , β and θ be 1, -1 and -2 . Then, the score of the alignment according to Equation (2.3) is -5 .

The scoring scheme provides a quantitative measure of how good an alignment is. In the literature, the problem of maximizing this score is known as the *sequence alignment problem*. Finding an optimal alignment between two sequences (*pairwise sequence alignment*) can be a computationally complex task as there is a considerable number of possible alignments. *Dynamic programming* (DP) is a method that allows solving this problem in an efficient manner, that is, in time proportional to the size of the sequences. DP is particularly well suited for this problem due to the natural definition of sub-problem that arises from the prefixes of the sequences. In the following subsections, we discuss different methods based on DP that have been proposed in the literature for pairwise sequence alignment.

2.3.3 Needleman-Wunsch Algorithm for Global Alignment

The Needleman-Wunsch Algorithm (Needleman and Wunsch, 1970) is a DP algorithm that solves the problem of sequence alignment. It was one of the first applications of DP to compare biological sequences. In this section, we explain this algorithm and show its correctness.

To determine the degree of similarity between two sequences, we use the score function as given in Definition 2.3.2. For finding the optimal alignment between two sequences, we need to compute the maximum score of this function and the alignment that yields it. We use the DP paradigm to solve it. A DP algorithm includes four parts:

- i) a recursive definition of the score;
- ii) a dynamic programming matrix for storing the optimal scores of sub-problems;
- iii) a bottom-up approach to complete the matrix starting from the smallest subproblems;

iv) a traceback method to recover the optimal alignment (Kleinberg and Tardos, 2006).

In the following, we describe these four parts in more details.

A Recursive Definition of the Score

The working principle for solving the pairwise sequence alignment problem with DP is to compute the optimal alignment for all pairs of prefixes of the given sequences. In the following, we explain the recursive formula for calculating the maximum score of the prefixes of the sequences. First, we give a well-known fundamental property of an alignment.

Lemma 2.3.1. (Kleinberg and Tardos, 2006)

Let $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$ be two sequences. An alignment φ of A and B does not contain any crossing pair, that is, if $(a_i, b_j), (a_{i'}, b_{j'}) \in \varphi$, and $i < i'$, $i, i' \in \{1, \dots, n_1\}$, $j, j' \in \{1, \dots, n_2\}$, then $j < j'$.

Proposition 2.3.2. Let φ be any alignment of $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$. If $(a_{n_1}, b_{n_2}) \notin \varphi$, then either the n_1 -th position of A or the n_2 -th position of B is aligned with character ‘-’.

Proof. This proof is done by contradiction: Suppose that b_{n_2} is aligned to $a_i \in A$, $1 \leq i < n_1$ in A and a_{n_1} is aligned to a position $b_j \in B$, $1 \leq j < n_2$ in B ; then, there exists a crossing pair, which contradicts Lemma 2.3.1. \square

The proposition states that an alignment between two sequences can end in one of the following three pairs: (a_{n_1}, b_{n_2}) , $(a_{n_1}, \text{'-'})$ or $(\text{'-'}, b_{n_2})$. The following corollary is equivalent to Proposition 2.3.2.

Corollary 2.3.3. (Barton and Sternberg, 1987) Let $\varphi = (\varphi_1, \dots, \varphi_\ell)$ be an optimal alignment with length ℓ of two sequences $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$. The alignment can possibly end in three ways:

- i) characters a_{n_1} and b_{n_2} are aligned to each other, i.e., $\varphi_\ell = (a_{n_1}, b_{n_2})$;
- ii) character a_{n_1} is aligned to an indel, i.e., $\varphi_\ell = (a_{n_1}, \text{'-'})$;
- iii) character b_{n_2} is aligned to an indel, i.e., $\varphi_\ell = (\text{'-'}, b_{n_2})$.

The corollary leads us directly to the formulation of the recurrence.

Let $\delta(i, j)$ denote the maximum score of an optimal alignment between the two prefixes (a_1, \dots, a_i) and (b_1, \dots, b_j) , $1 \leq i < n_1$, $1 \leq j < n_2$. Let M be a scoring matrix and let θ be the indel score for each character of A or B that is aligned to an indel character. If case i) holds, we have that

$$\delta(n_1, n_2) = M[a_{n_1}, b_{n_2}] + \delta(n_1 - 1, n_2 - 1)$$

If case ii) and iii) holds we have that

$$\delta(n_1 - 1, n_2) = \theta + \delta(n_1 - 1, n_2) \quad \text{and} \quad \delta(n_1, n_2 - 1) = \theta + \delta(n_1, n_2 - 1)$$

respectively. As a result, the optimal alignment is the highest scoring of these three cases.

Before writing the general recursive formula, we need to show that, if there exists an optimal alignment for the two sequences, by removing one of the three alternative pairs from the optimal alignment, the remaining alignment is also optimal for the three alternative prefixes. This shows that *optimal substructure* exists, which is an important property of an optimization problem that can be explored by DP.

Proposition 2.3.4. *Let $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$ be two sequences. Let $\varphi = (\varphi_1, \dots, \varphi_\ell)$ be an optimal alignment of A and B with maximum score $\delta(n_1, n_2)$.*

- i) If $\varphi_\ell = (a_{n_1}, b_{n_2})$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is an optimal alignment of (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{n_2-1}) ;*
- ii) if $\varphi_\ell = (a_{n_1}, ' -')$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is an optimal alignment of (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{n_2}) ;*
- iii) if $\varphi_\ell = (' - ', b_{n_2})$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is an optimal alignment of (a_1, \dots, a_{n_1}) and (b_1, \dots, b_{n_2-1}) .*

Proof.

- i) Assume that $(\varphi_1, \dots, \varphi_{\ell-1})$ is not an optimal alignment of prefix (a_1, \dots, a_{n_1-1}) and prefix (b_1, \dots, b_{n_2-1}) . Then there exists another optimal alignment with larger score for the same prefixes. By summing this score to $M[a_{n_1}, b_{n_2}]$, we obtain an alignment with larger score than $\delta(n_1, n_2)$, which is a contradiction;
- ii) Assume that $(\varphi_1, \dots, \varphi_{\ell-1})$ is not an optimal alignment of prefix (a_1, \dots, a_{n_1-1}) and prefix (b_1, \dots, b_{n_2}) . Then there exists another optimal alignment with larger score for the same prefixes. By summing up θ to the score of that alignment, we obtain an alignment with a larger score than $\delta(n_1, n_2)$, which is a contradiction;

iii) Symmetric to ii).

□

By applying the same argument inductively for the subproblem of finding the maximum score alignment between (a_1, \dots, a_i) and (b_1, \dots, b_j) , $1 \leq i < n_1$, $1 \leq j \leq n_2$, we arrive to the following general recursive definition.

$$\delta(i, j) = \begin{cases} i \cdot \theta & \text{if } j = 0 \\ j \cdot \theta & \text{if } i = 0 \\ \max \begin{cases} \delta(i-1, j-1) + M[a_i, b_j] \\ \delta(i-1, j) + \theta \\ \delta(i, j-1) + \theta \end{cases} & \text{if } i \neq 0 \text{ and } j \neq 0 \end{cases} \quad (2.4)$$

Where $\delta(0, j)$ and $\delta(i, 0)$ correspond to the base case of the recursion, that is, the score of aligning a prefix with an empty sequence. Due to the *overlapping subproblem* property of this problem, it is possible to store the values of $\delta(i, j)$ in a matrix to discard recursive function calls, which would correspond to a *top-down* DP approach. In order to avoid the overhead of recursion, we introduce the *bottom-up* version in the following section.

Dynamic Programming Matrix and Bottom-up Approach

We consider a two-dimensional matrix P with the size $(n_1+1) \times (n_2+1)$, where the value $P[i, j]$ is the score of the optimal alignment with the prefixes (a_1, a_2, \dots, a_i) and (b_1, b_2, \dots, b_j) , i.e., $\delta(i, j)$. Matrix $P[i, j]$ is built in two steps. (Needleman and Wunsch, 1970)

1. *Initialization.* This step treats the base case of the recursion in Eq. (2.4). The $[0, 0]$ element of P is set to 0 and the top row and leftmost column of the matrix are initialized with the cost of indels of lengths i and j , respectively.
2. *Matrix fill.* The matrix is filled row-wise from top-left to bottom-right. There are three possible ways that the best score of an alignment up to a_i and b_j , $P[i, j]$, $1 \leq i \leq n_1$, $1 \leq j \leq n_2$, is obtained: a_i could be aligned to b_j , $P[i, j] = P[i-1, j-1] + M[a_i, b_j]$; or a_i is aligned to an indel character, $P[i, j] = P[i-1, j] + \theta$; or b_j is aligned to an indel, $P[i, j] = P[i, j-1] + \theta$. The best score up to (i, j) will be the maximum of these three options.

		0	1	2	3	4
		-	A	G	T	A
0	-	0	-2	-4	-6	-8
1	A	-2	↖ 1	← -1	-3	-5
2	T	-4	-1	0	↖ 0	-2
3	A	-6	-3	-2	-1	↖ 1

Figure 2.2: Illustration of the DP matrix of the Needleman-Wunsch algorithm and the traceback of the alignment (red arrows)

$$P[i, j] = \max \begin{cases} P[i-1, j-1] + M[a_i, b_j] \\ P[i-1, j] + \theta \\ P[i, j-1] + \theta \end{cases} \quad (2.5)$$

For nucleotides sequences, matches are rewarded with α , mismatches are penalized by β and indels are penalized by θ . The corresponding recurrence can be rewritten as:

$$P[i, j] = \max \begin{cases} P[i-1, j-1] + \alpha, & \text{if } a_i = b_j \\ P[i-1, j-1] + \beta, & \text{if } a_i \neq b_j \\ P[i-1, j] + \theta \\ P[i, j-1] + \theta \end{cases} \quad (2.6)$$

The recurrence is repeatedly applied to fill the matrix P and once it is filled, the last element of the matrix at position (n_1, n_2) , holds the score of the optimal alignment.

The Traceback Method

In order to reconstruct an optimal alignment, and given that matrix P is filled, there is only the need to trace back the decisions taken by the DP algorithm above.

Figure 2.2 shows the matrix P for the nucleotide sequences AGTA and ATA with match reward 1 ($\alpha = 1$), mismatch penalty -1 ($\beta = -1$) and indel penalty ($\theta = -2$). From the bottom-right cell, a route is traced back to the top-left cell, which gives the alignment. By following the path in the matrix, an optimal alignment, with score 1, is:

AGTA

A-TA

One may be interested in reconstructing all optimal alignments. In that case, each of the three options that are selected during the matrix fill needs to be stored in each cell. Note that if multiple options result in the same score, all of them need to be stored. This can be performed by storing ‘predecessor’ pointers for each cell, or by storing the result for each of the three options in a separate matrix and looking up which has the highest result for a cell.

2.3.4 Global Alignment with Gap Penalties

Defining the best alignment between two sequences depends on the score function. It is possible that for two different score functions the best alignments differ considerably. Therefore, it is crucial that a score function is biologically relevant in order to produce an alignment that may have a biological interpretation (Jones and Pevzner, 2004). Such a function must take into account factors that constrain sequence evolution. The notion of a gap in an alignment is one of these factors.

Usually, a gap is penalized according to its length in the alignment. However, clearly, the way gaps are scored critically influences the effectiveness of the gap. There are different models of gap penalties such as constant, linear, affine, convex, and arbitrary (Gusfield et al., 1994). The first three gap models are as follows.

- *Constant gap model.* This model is the simplest type of calculating gap penalty, for which each gap is scored independently of its length. Consider the following alignment of two short DNA sequences:

$$\begin{array}{cccccc}
 \text{G} & \text{T} & - & - & - & \text{C} \\
 \text{G} & \text{T} & \text{A} & \text{A} & \text{G} & \text{C} \\
 +1 & +1 & & -1 & & +1
 \end{array}$$

Assuming that each match is 1 and each gap is -1 , the total score of the alignment under the constant gap model is $1 + 1 + 1 - 1 = 2$.

- *Linear gap model.* This model takes into account the length of the gap. In this case, each indel in the gap is penalized with a fixed negative value. The problem of finding the optimal alignment with gap penalties by considering linear gap model is the same as given in Eq. (2.3).

$$\begin{array}{cccccc}
 \text{G} & \text{T} & - & - & - & \text{C} \\
 \text{G} & \text{T} & \text{A} & \text{A} & \text{G} & \text{C} \\
 +1 & +1 & -1 & -1 & -1 & +1
 \end{array}$$

Assuming that each indel is penalized by -1 , then the total score of the alignment by considering the linear gap model is $1 + 1 + 1 - 1 - 1 - 1 = 0$.

- *Affine gap model.* This is the most widely used model. The score of a gap region of length g is equal to $\nu + \mu(g - 1)$, where ν and μ are fixed negative values. In this case, ν is called the *gap open weight*, since it represents the cost of starting a gap, and μ is called the *gap extension weight*. Usually, μ is set to be smaller than ν , which reflects the fact known from biology that starting a gap region is harder than extending it (Gusfield et al., 1994).

G	T	-	-	-	C
G	T	A	A	G	C
+1	+1	-5	-1	-1	+1

Assuming that the gap open weight is -5 and the gap extension weight is -1 , the score of the alignment by considering the affine gap model is $3 - 5 - 1 - 1 = -4$.

The Needleman-Wunsch Algorithm (Needleman and Wunsch, 1970) solves the problem of pairwise sequence alignment with the linear gap model (see Section 2.3.3). In the following, we describe an extension of this algorithm for the affine gap model (Gusfield et al., 1994).

Definition 2.3.3. *Score of an alignment with affine gap model.*

Let φ be an alignment of two sequences $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$. Let α, β, ν, μ denote weights for match, mismatch, gap open weight and gap extension weight, respectively. Let $m(\varphi)$, $ms(\varphi)$, $g(\varphi)$ and $d(\varphi)$ indicate the number of matches, mismatches, gaps and indels orderly in the alignment φ . Then, the score function of an alignment φ under the affine gap model is computed as follows:

$$\delta(\varphi) = \alpha \cdot m(\varphi) + \beta \cdot ms(\varphi) + \nu \cdot g(\varphi) + \mu \cdot (d(\varphi) - g(\varphi)).$$

The DP algorithm that finds the alignment that is maximal for this score function is discussed in Gusfield et al. (1994). It requires four matrices: Q , R , S , and T . For a given $(i, j) \in \{1, \dots, n_1\} \times \{1, \dots, n_2\}$, entry $R[i, j]$, $S[i, j]$ and $T[i, j]$ stores the set of states corresponding to optimal alignments of prefixes (a_1, \dots, a_i) and (b_1, \dots, b_j) that end with (a_i, b_j) , $(\text{'-'}, b_j)$ and $(a_i, \text{'-'})$, respectively, $1 \leq i \leq n_1$, $1 \leq j \leq n_2$. The element $Q[i, j]$ is the score corresponding to optimal alignment of prefixes (a_1, \dots, a_i) and (b_1, \dots, b_j) . Then, the recursion by considering a substitution matrix M , gap open weight ν and gap extension weight μ is as follows:

$$Q[i, j] = \max \begin{cases} R[i, j] \\ S[i, j] \\ T[i, j] \end{cases}$$

$$R[i, j] = Q[i - 1, j - 1] + M[a_i, b_j]$$

$$S[i, j] = \max \begin{cases} S[i, j - 1] + \mu \\ Q[i, j - 1] + \nu \end{cases}$$

$$T[i, j] = \max \begin{cases} T[i - 1, j] + \mu \\ Q[i - 1, j] + \nu \end{cases}$$

with the following bases cases

$$Q[0, 0] = 0$$

$$Q[i, 0] = T[i, 0] = \nu + (i - 1) \cdot \mu$$

$$Q[0, j] = S[0, j] = \nu + (j - 1) \cdot \mu$$

with $1 \leq i \leq n_1$, $1 \leq j \leq n_2$. A bottom-up DP algorithm can easily be derived from the above recursions, analogously to the Needleman-Wunsch algorithm described in Section 2.3.3.

2.3.5 Smith-Waterman Algorithm for Local Alignment

Global alignment methods force alignments to span the entire length of the sequences by attempting to align every character of each sequence. They are useful when two sequences are similar and have roughly equal length. Local alignment methods seek to identify regions of similarity between two sequences. Local alignments are more useful to identify discrete regions of similarity between otherwise divergent sequences; for example, when it is suspected that two protein sequences share a common domain, or when comparing extended sections of genomic DNA sequence. It is also the most sensitive way to detect similarity when comparing two very highly diverged sequences, even when they may have a shared evolutionary origin along with their entire length (Smith and Waterman, 1981).

The algorithm for finding an optimal local alignment is called the Smith-Waterman algorithm (Smith and Waterman, 1981). It is closely related to the algorithm for global alignment as described in the previous section. The two main differences are:

- i) In each cell in the matrix, $P[i, j]$ can take the value zero if all the three options have negative values. This option allows the algorithm to start a new alignment, which might result in a better score of the prefixes' alignment than continuing an existing one. This translates into resetting the current score of the prefixes' alignment to zero.
- ii) In order to reconstruct the alignment, the traceback must start from the highest value in P . The traceback ends when a cell with value zero is found.

The recurrence equation for local alignment is given as follows:

$$P[i, j] = \max \begin{cases} 0 & \text{for } i \geq 0 \text{ or } j \geq 0, \\ P[i - 1, j] + \theta & \text{for } i > 0, \\ P[i, j - 1] + \theta & \text{for } j > 0, \\ P[i - 1, j - 1] + M[a_i, b_j] & \text{for } i > 0 \text{ or } j > 0. \end{cases} \quad (2.7)$$

with $1 \leq i \leq n_1$, $1 \leq j \leq n_2$. A bottom-up DP algorithm can also be derived from the above recursions, analogously to the Needleman-Wunsch algorithm described in Section 2.3.3.

A comparison between global and local alignment of two hypothetical genes is shown in Figure 2.3 (Jones and Pevzner, 2004, page 182). By considering the scoring scheme with match reward 1 ($\alpha = 1$), mismatch penalty -1 ($\beta = -1$) and indel penalty ($\theta = -2$); the alignment score of the global alignment is -29 whereas for the local alignment is 12.

2.4 Multiple Sequence Alignment Problem

A multiple sequence alignment (MSA) is an alignment of more than two biological sequences. Usually, the set of sequences considered in this problem is assumed to have some evolutionary relationship. From the resulting MSA, sequence homology can be inferred, and a phylogenetic tree can be conducted to assess the shared evolutionary origins of the sequences.

This section introduces the problem of finding an alignment for more than two sequences. First, the required mathematical definition and three different models of scoring an MSA are presented. Then, the strategies for solving this problem are introduced. An exact algorithm, which is a natural extension of the DP algorithm presented in Section 2.3, is discussed as well as heuristic methods such as progressive alignments and iterative approaches.

2.4.1 Notation and Definitions

Multiple alignment can formally be defined analogously to the alignment of two sequences as follows.

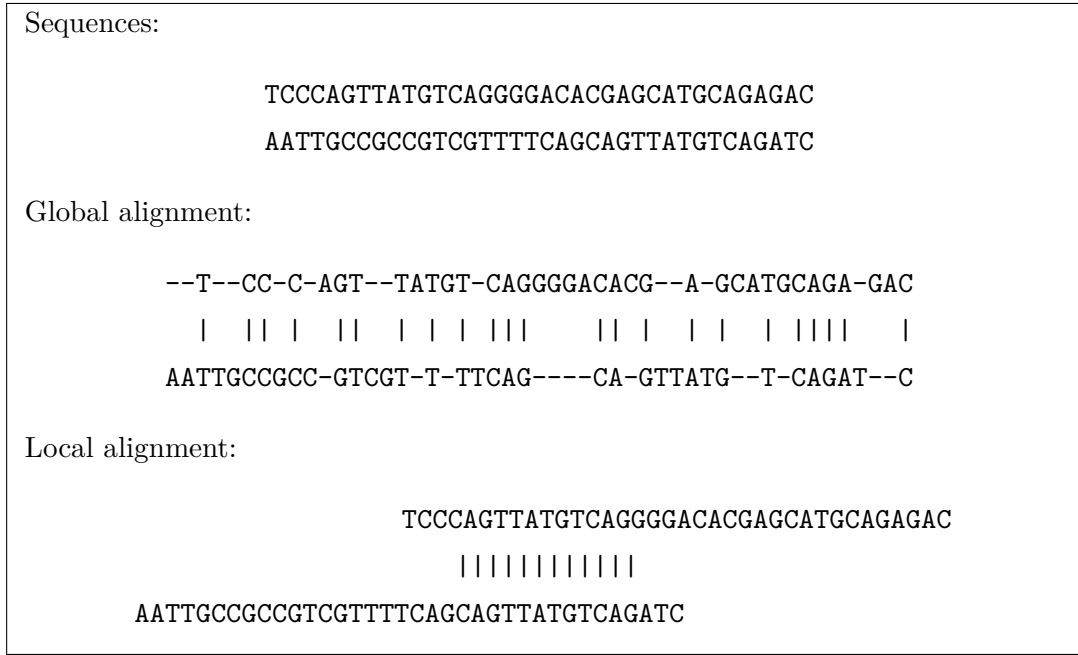


Figure 2.3: A comparison of the global versus local alignment (Jones and Pevzner, 2004, page 182)

Definition 2.4.1. *Multiple sequence alignment (Bockenhauer and Bongartz, 2007, page 101).*

Let $A_1 = (a_{11}, \dots, a_{1n_1}), \dots, A_m = (a_{m1}, \dots, a_{mn_m})$ be m sequences over an alphabet Σ . Let $'-'$ $\notin \Sigma$ be an indel character and let $\Sigma' = \Sigma \cup \{'-\'}$ and λ denotes the empty string. Let $h : (\Sigma')^* \rightarrow \Sigma^*$ be a homomorphism defined by $h(a) = a$ for all $a \in \Sigma$, and $h('-') = \lambda$. A multiple sequence alignment ϕ of (A_1, \dots, A_m) is an m -tuple of $A'_1 = (a'_{11}, \dots, a'_{1\ell}), \dots, A'_m = (a'_{m1}, \dots, a'_{m\ell})$ of sequences with length ℓ such that $\ell \geq \max\{|A_i|, 1 \leq i \leq m\}$ over the alphabet Σ' , such that the following conditions are satisfied:

- i) $|A'_1| = |A'_2| = \dots = |A'_m|$;
- ii) $h(A'_i) = A_i$ for all $i \in \{1, \dots, m\}$;
- iii) For every $j \in \{1, \dots, \ell\}$ there exists an $i \in \{1, \dots, m\}$ such that $a'_{ij} \neq '-'$.

Note that Condition iii) of the definition above indicates that it does not exist any $j \in \{1, \dots, \ell\}$ such that an indel character occurs in all a'_{ij} , $i \in \{1, \dots, m\}$.

There are different ways of scoring an MSA, which may lead to different multiple sequence alignments. In the following, the three most known score functions are described in detail.

i) *Sum-of-pairs score.*

The most widely used scoring scheme for MSA is the sum of pairs score (SP-score), which consists of summing up the pairwise sequence alignment score between each pair of sequences in the alignment. The SP-score $\delta_{\text{SP}}(\phi)$ of an MSA ϕ is computed as follows:

$$\delta_{\text{SP}}(\phi) = \sum_{1 \leq i < j \leq m} \delta(\phi^{ij})$$

where ϕ^{ij} denotes the pairwise alignment between sequences A_i and A_j and $\delta(\phi^{ij})$ is the pairwise alignment score as given in Definition 2.3.3. By considering the affine gap model, we can recast the score function above as follows.

$$\delta(\phi^{ij}) = \sum_{k=1}^{\ell} s(a'_{ik}, a'_{jk})$$

where

$$s(a, b) = \begin{cases} M[a, b] & \text{if } (a \neq '-' \text{ and } b \neq '-') \\ \nu & \text{if } (a \neq '-' \text{ xor } b \neq '-') \text{ and it is a gap opening} \\ \mu & \text{if } (a \neq '-' \text{ xor } b \neq '-') \text{ and it is a gap extension} \\ 0 & \text{if } (a = '-' \text{ and } b = '-') \end{cases}$$

and ℓ is the length of ϕ . Function $s(a, b)$ evaluates each pair of aligned residues in the MSA. The score for the matching of residues a and b is obtained from the substitution matrix M . If either a or b is an indel character and it is a gap opening, then $s(a, b)$ is equal to ν (gap opening penalty), otherwise if it is a gap extension then the score is equal to μ (gap extension penalty). If both are '-', then $s(a, b) = 0$.

If the linear gap model (see Section 2.3.4 page 21) with a fixed indel penalty μ is used, this formulation can be simplified as follows.

$$\delta_{\text{SP}}(\phi) = \sum_{k=1}^{\ell} \sum_{i=1}^{m-1} \sum_{j=i+1}^m s(a'_{ik}, a'_{jk})$$

where

$$s(a, b) = \begin{cases} M[a, b] & \text{if } (a \neq '-' \text{ and } b \neq '-') \\ \mu & \text{if } (a \neq '-' \text{ xor } b \neq '-') \\ 0 & \text{if } (a = '-' \text{ and } b = '-') \end{cases}$$

In the following, we provide an example for the calculation of this score function.

Example 2.2. Consider the following MSA of the sequences TACCAGT, CCCGTAA and TCC:

	1	2	3	4	5	6	7	8	9
1	T	A	C	C	A	G	T	-	-
2	C	-	C	C	-	G	T	A	A
3	T	-	C	C	-	-	-	-	-

Let the score of (a, b) be 1, if $a = b$ and 0 otherwise. Let the penalty score be $\mu = -1$. Then, the SP-score of this alignment is as follows:

$$\begin{aligned} \delta_{SP}(\phi) &= \sum_{k=1}^9 \sum_{i=1}^2 \sum_{j=i+1}^3 s(a'_{ik}, a'_{jk}) \\ &= (0 + 1 + 0) + (-1 - 1 + 0) + (1 + 1 + 1) + (1 + 1 + 1) + (-1 - 1 + 0) \\ &\quad + (-1 + 1 - 1) + (-1 + 1 - 1) + (-1 + 0 - 1) + (-1 + 0 - 1) \\ &= -3 \end{aligned}$$

ii) *Weighted Sum-of-Pairs score.*

The weighted sum-of-pairs score (WSP-score) is an extension of SP-score, where each pair of sequences contributes differently to the total score. The WSP-score is defined as:

$$\delta_{WSP}(\phi) = \sum_{1 \leq i < j \leq m} \omega_{i,j} \cdot \delta(\phi^{ij})$$

where $\omega_{i,j}$ is the weight associated with the pair of sequences A_i and A_j in the alignment. This weight is usually calculated according to the distances between the two

sequences using methods such as UPGMA (Sneath et al., 1975) or Neighbor Joining (Saitou and Nei, 1987) (see Section 6.3.2, page 135).

iii) *Maximum consistency score.*

The maximum consistency score is used by recent MSA programs such as T-Coffee (Notredame et al., 2000), MAFFT version 5 (Kato et al., 2002) and ProbCons (Do et al., 2005). For calculating the score of an alignment with this method, the substitution score is calculated between each residue pair using the information from pairwise alignments (local and global) between sequences. This method is known to increase the accuracy of the MSA, but it has a costly computation time (Notredame et al., 1998). The calculation of this score is explained more in detail in the following section (page 33).

2.4.2 Algorithms for the MSA Problem

In the following, we introduce the most common strategies for solving the MSA problem. The algorithms are classified into three main categories: exact, progressive and iterative algorithms (Notredame, 2002). The exact algorithm solves the MSA problem with a DP approach using an n -dimensional matrix from n sequences. Unfortunately, this method needs a significant amount of time and memory resources and is not feasible in practice.

The most commonly used heuristic methods are progressive algorithms. They build up the MSA by performing a series of pairwise alignments. This approach starts with a particular pairwise alignment and iteratively adds a sequence or a pairwise alignment to the growing multiple alignment. One of the primary problems of progressive alignments is that, when errors are made at any stage in the growing MSA, they are propagated through the final result. Iterative algorithms are similar to progressive methods, but repeatedly realign the initial sequences and add new sequences to the growing MSA in order to refine it.

Exact Algorithms

The MSA problem can be solved by extending the DP approach for the pairwise sequence alignment problem (see Section 2.3.3). By assuming an SP-scoring method with a linear gap cost function (see Section 2.4.1) the following parameters are required. For calculating the SP-score (δ_{SP}), we assume that the scores $s(a, b)$ are available from a substitution matrix M , $s(a, '-')$ is a fixed value (indel weight) and $s('-', '-') = 0$. Suppose we have m sequences $A_1 = (a_{11}, \dots, a_{1n_1}), A_2 = (a_{21}, \dots, a_{2n_2}), \dots, A_m = (a_{m1}, \dots, a_{mn_m})$. Let $P[i_1, \dots, i_m]$ be

the maximal score of an alignment for the prefixes ending with $a_{1i_1}, \dots, a_{mi_m}$. The recursion step of the dynamic programming algorithm is given by the following recurrence (Jones and Pevzner, 2004; Cohen, 2004; Isaev, 2004):

$$P[i_1, \dots, i_m] = \max \left\{ \begin{array}{l} P[i_1 - 1, \dots, i_m - 1] + \delta_{SP}(a_{1i_1}, \dots, a_{mi_m}) \\ P[i_1, i_2 - 1, \dots, i_m - 1] + \delta_{SP}(\text{' - '}, a_{2i_2}, \dots, a_{mi_m}) \\ P[i_1 - 1, i_2, i_3 - 1, \dots, i_m - 1] + \delta_{SP}(a_{1i_1}, \text{' - '}, a_{3i_3}, \dots, a_{mi_m}) \\ \vdots \\ P[i_1 - 1, \dots, i_{m-1} - 1, i_m] + \delta_{SP}(a_{1i_1}, \dots, a_{m-1i_{m-1}}, \text{' - '}) \\ \vdots \\ P[i_1, i_2, i_3 - 1, \dots, i_m - 1] + \delta_{SP}(\text{' - '}, \text{' - '}, a_{3i_3}, \dots, a_{mi_m}) \\ \vdots \\ P[i_1, i_2, i_3, \dots, i_m - 1] + \delta_{SP}(\text{' - '}, \text{' - '}, \dots, \text{' - '}, a_{mi_m}) \end{array} \right.$$

where all combination of indels occurs except the one where all residues are replaced by indels. The algorithm is initialized by setting $P[0, \dots, 0] = 0$. The traceback starts at $P[n_1, \dots, n_m]$ and is analogous to the traceback method for pairwise alignments (see Section 2.3).

For m sequences, the method above requires constructing an m -dimensional matrix equivalent to the one for pairwise sequence alignment (see Section 2.3). Therefore, the search space increases exponentially with increasing m and is also strongly dependent on the length of the sequences. If the length of the largest sequence is equal to ℓ , then the space and time complexity of this method would be $O(\ell^m)$. In fact, the problem of finding the global optimum for an arbitrary number of sequences has been shown to be NP-hard (Just, 2001; Isaac, 2006). Carrillo and Lipman introduced some speed-up technique that uses pairwise alignments to constrain the m -dimensional search space. The idea is to reduce the number of elements of the DP matrix that need to be examined in order to find an optimal multiple alignment (Carrillo and Lipman, 1988; Lipman et al., 1989). Still, the exact algorithm has large memory and computational time. Thus, it is impractical for many MSA applications that require the simultaneous alignment of many sequences.

Progressive Algorithms

Progressive algorithms are heuristic approaches to MSA (Jones and Pevzner, 2004). Heuristics are applied to problems where the time required to find an optimal solution is very large. These approaches do not guarantee the optimal solution but employ clever techniques to find a satisfactory solution in a short amount of time.

Progressive alignment algorithms produce a multiple sequence alignment from a number of pairwise alignments. They greedily select a pair of sequences with the largest similarity score and merges it into a new sequence or already constructed pairwise alignment following the principle “once a gap, always a gap”. This principle states that once a gap is introduced into the alignment, it will never be removed, even if that leads to a worse multiple alignment (Feng and Doolittle, 1987). The motivation for the choice of the most similar sequences in the early steps of the algorithm is that they often provide the most reliable information about the optimal alignment. Clearly, different alignments can be produced by considering different orderings of the sequences to be merged.

In the following, we discuss two progressive algorithms that have been used extensively by practitioners, ClustalW (Thompson et al., 1994) and T-Coffee (Notredame et al., 2000).

ClustalW One of the most popular multiple sequence alignment algorithms that uses a progressive strategy is ClustalW (Thompson et al., 1994). The three basic steps in the ClustalW approach are almost equal to all progressive alignment algorithms: i) Calculate a matrix of pairwise distances based on pairwise alignments between the sequences; ii) Use the result of (i) to build a guide tree, which is an inferred phylogeny for the sequences (see Section 6.3.2); iii) Use the tree from (ii) to guide the progressive alignment of the sequences. We illustrate this technique by considering the following four sequences of protein sequences:

```
SeqA:  KNLFMELD TPE LNST FNT RNT
SeqB:  KNLFMELD TPE FNST RNT
SeqC:  KNLFMELD TPE AELY FNST RNT
SeqD:  TPE FNT RNT
```

i) Calculation of the pairwise distances

In the first step, all pairs of sequences are aligned by a DP algorithm, and then a similarity or distance value is calculated for each pair using the aligned portion (indels/gaps are excluded). Figure 2.4 shows an optimal pairwise alignment of all pairs of the four


```

SeqA: KNLFMELDTPELNSTFNTRNT      SeqB: KNLFMELDTPE----FNSTRNT
SeqB: KNLFMELDTPEFNSTRNT---      SeqC: KNLFMELDTPEAELYFNSTRNT

SeqA: KNLFMELDTPELNSTFN--TRNT    SeqB: KNLFMELDTPEFNSTRNT
SeqC: KNLFMELDTPEAELYFNSTRNT      SeqD: -----TPEFN--TRNT

SeqA: KNLFMELDTPELNSTFNTRNT      SeqC: KNLFMELDTPEAELYFNSTRNT
SeqD: -----TPEFN----TRNT        SeqD: -----TPEFN----TRNT

```

Figure 2.4: All pairwise alignments for the four sequences²

sequences above. A possible measure of distance is to use the number of matches in the alignment as follows (Thompson et al., 1994):

$$distance = 1 - \frac{\text{Number of matches}}{\text{Length of aligned region with no indels/gaps}}$$

In our example, the distance between sequences A and B is equal to $1 - 16/18 = 0.1(1)$. Note that we have computed distances to be in the range of 0 to 1, with smaller values indicating more closely related sequences. The distance matrix for the sequences of the example is shown in the following table.

	Seq.A	Seq B	Seq C	Seq D
Seq A	0.0000	0.1(1)	0.1905	0.1(1)
Seq B	0.1(1)	0.0000	0.0000	0.0000
Seq C	0.1905	0.0000	0.0000	0.2(2)
Seq D	0.1(1)	0.0000	0.2(2)	0.0000

ii) Building the guide tree

In the following step, the guide tree is built based on the distance matrix computed in the previous step. The exact details of the tree construction are discussed in Chapter 6. ClustalW uses the neighbor-joining method (Saitou and Nei, 1987) to compute this tree. The sum of the branch lengths of the guide tree between each pair of sequences corresponds, ideally, to the distance between those sequences. However, the current methods build a tree that approximate the information given in the distance matrix.

²The global pairwise alignments were obtained from the online tool EMBOSS Needle, available at http://www.ebi.ac.uk/Tools/psa/emboss_needle, with substitution matrix BLOSUM62, gap opening value -10 and gap extension weight -0.5.

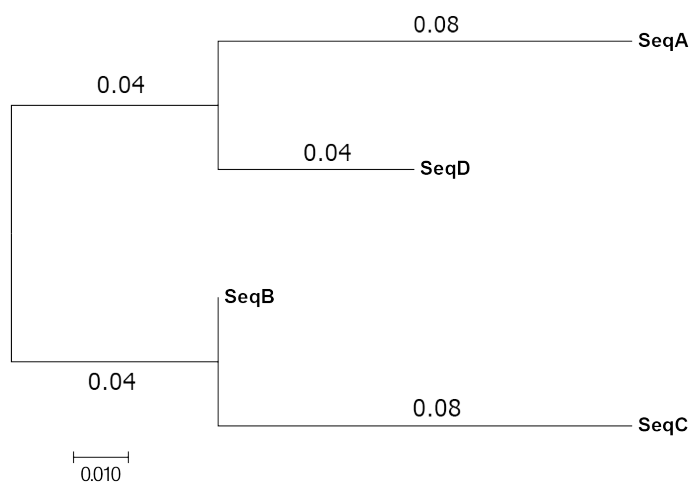


Figure 2.5: The guide tree obtained with the Neighbor-Joining method.³

Figure 2.5 shows the corresponding guide tree from the above distance matrix. As it is shown, the distance between SeqA and SeqD in the tree is $0.08 + 0.04 = 0.12$ which is close to the value 0.1(1) in the distance matrix.

iii) *Progressive alignment*

In this step, the idea is to use the pairwise alignments already computed to align larger and larger groups of sequences, following the branching order of the guide tree. Alignments are combined starting from the most closely related sequences and then proceed from the leaves of the tree towards the root. Each step consists of aligning two existing alignments or a sequence with alignments. At each stage, a DP algorithm is used with a substitution matrix (e.g., a PAM or a BLOSUM matrix) and gap opening and extension penalties. The score at each position in the DP matrix is equal to the average of all scores from the two sets of sequences. We illustrate this procedure by considering the two following pairwise alignments:

```

seqB:  KNLFMELDTPE----FNSTRNT
seqC:  KNLFMELDTPEAELYFNSTRNT

seqA:  KNLFMELDTPELNSTFNTRNT
seqD:  -----TPEFN----TRNT

```

³The tree was generated using MEGA7 (Kumar et al., 2016) available at <http://www.megasoftware.net/>

The scoring of the first column of the first alignment against the fourth column of the second alignment is calculated as follows (given a substitution matrix M):

$$(2 \cdot M[\text{K}, \text{K}] + 2 \cdot M[\text{F}, \text{' - '}] / 4$$

For the four sequences of the example, the alignment order is the following:

- i) Align SeqB with SeqC (Group 1)

```
KNLFMELDTPE----FNSTRNT
KNLFMELDTPEAELYFNSTRNT
```

- ii) Align SeqA and SeqD (Group 2)

```
KNLFMELDTPELNSTFNTRNT
-----TPEFN-----TRNT
```

- iii) Align Group 1 with Group 2

```
KNLFMELDTPE----FNSTRNT
KNLFMELDTPEAELYFNSTRNT
KNLFMELDTPELNSTFNTRNT-
-----TPEFN-----TRNT
```

T-Coffee This progressive method for MSA uses *consistency information* for the scoring (Notredame et al., 2000). The main idea of this technique is to preprocess a dataset of all pairwise alignments between the sequences, which provides a library of alignment information that is later used to guide the progressive strategy. We illustrate the steps of this technique with the four sequences used in the previous example.

- i) *Generation of the primary library*

The primary library is created by generating the pairwise alignments between all pairs of sequences. For each pair of sequences, T-Coffee calculates one global pairwise alignment and several non-overlapping local alignments. The global alignments are usually constructed by using the ClustalW program and default parameters. The local alignments are generated by Lalign program⁴ with default parameter (Huang and Miller, 1991). The algorithm for each sequence pair chooses the top ten scoring local alignments that do not have intersections.

⁴Lalign finds a set of non-overlapping local alignments (Huang and Miller, 1991; Pearson and Lipman, 1988). It is an implementation of Sim program inside FASTA package (see http://embnet.vital-it.ch/software/LALIGN_form.html)

SeqA: KNLFMELDTPELNSTFNTRNT	PW=88	SeqB) KNLFMELDTPE----FNSTRNT	PW=100
SeqB: KNLFMELDTPEFNSTRNT---		SeqC: KNLFMELDTPEAELYFNSTRNT	
SeqA: KNLFMELDTPELNSTFN-TRNT	PW=77	SeqB: KNLFMELDTPEFNSTRNT	PW=100
SeqC: KNLFMELDTPEAELYFNSTRNT		SeqD: -----TPEFN-TRNT	
SeqA: KNLFMELDTPELNSTFNTRNT	PW=100	SeqC: KNLFMELDTPEAELYFNSTRNT	PW=100
SeqD: -----TPE----FNTRNT		SeqD: -----TPE----FN-TRNT	

Figure 2.6: T-Coffee primary library, with all pairwise alignments and the Primary Weight value (PW)

ii) *Calculation of the primary weights*

In the primary library, the alignments are kept as a list of aligned residue pairs, for example, if the residue x of sequence A is aligned with the residue y of sequence B , then the alignment contains the residue pair $(A(x), B(y))$. These residue pairs called *constraints*; Each constraint gets a value. In this method, to evaluate the multiple sequence alignment instead of using the values of substitution matrices, these values are considered. Nevertheless, all of these pairs may not have equal importance in producing the sequence alignment. Some of them may come from parts of alignment that seem to be more correct. Therefore, T-Coffee assigns a weight to each residue pair. The primary weight in the primary library is based on the sequence identity. In this step, each constraint (residue pair) get a value (weight) equal to the percentage of sequence identity of the pairwise alignment that it originates. Figure 2.6 shows all global alignments of the four sequences and the corresponding primary weights (PW).

iii) *Merging of the two primary libraries*

The primary library consists of local alignment and global alignment for a pair of sequences. Since a residue pair in the global alignment can appear in local alignments, the weights for such aligned residue pairs need to be merged. This merging step consists only of summing up the two weights if a residue pair is duplicated between the two libraries. Otherwise, the residue pair is stored as it is. For residue pairs that do not occur in any pairwise alignment, the default weight value is equal to zero.

iv) *Extending the merged primary library*

The merged primary weights can be used directly to compute the MSA, but T-Coffee

```

SeqA: KNLFMELD TPE LNST FNT RNT    weight = 88
SeqB: KNLFMELD TPE FNST RNT  ---

SeqA: KNLFMELD TPE LNST FN-T RNT
SeqC: KNLFMELD TPE AELY FNST RNT    weight = 77
SeqB: KNLFMELD TPE FNST RNT-  ---

SeqA: KNLFMELD TPE LNST FNT RNT
SeqD: ----- TPE ---- FNT RNT    weight = 96
SeqB: KNLFMELD TPE FNST RNT  ---

```

Figure 2.7: The three possible alignments of SeqA and SeqB for our example

extends this library information. It increases the value of the library information by testing the *consistency* of each residue pair with residue pairs from all the other alignments. For this purpose, T-Coffee takes each aligned residue pair from the primary library and check if it aligns with the residues from the remaining sequences. This procedure is illustrated in Figure 2.7. For the sake of the explanation, let a_K , b_K and c_K denote the index of the first K in SeqA, SeqB and SeqC, respectively and let $W(b_K, a_K)$ denote the weight of the pair (a_K, b_K) , which is 88. In the alignment of SeqA and SeqB through sequence SeqC, a_K and b_K are aligned, as well as b_K and c_K . Therefore, the residue pair (a_K, b_K) is aligned through SeqC. In order to increase the quality information of residue pair (a_K, b_K) of SeqA and SeqB, T-Coffee adds a weight equal to the minimum of $W(a_K, c_K)$ and $W(b_K, c_K)$, that is $\min(77, 100) = 77$, to its primary weight. Therefore, the weight of residue pair (a_K, b_K) increase to $88+77$. T-Coffee repeats this procedure for all the other triples. The algorithm converts consistency information into weights for aligned residue pairs. Once weights are updated, T-Coffee proceeds to the next step.

v) *Progressive strategy*

In the last step, T-Coffee progressively aligns sequences. This process occurs the same as the ClustalW procedure. Pairwise alignments are first calculated to create a distance matrix. Then, based on a distance matrix, a guide tree using the neighbor-joining method is calculated (Saitou and Nei, 1987). At last, with the help of a phylogenetic tree, the sequences align progressively to produce the MSA. The only difference is that for aligning in the last step, T-Coffee uses the weights from the extended library and

the gap penalties are set to zero.

Iterative Algorithms

One of the main problems with progressive alignments is that wrong choice occurring in the initial alignment will be propagated to the rest of the alignment procedure. The iterative approaches were introduced to overcome this problem. They repeatedly realign all sequences so that the overall alignment score, such as the sum-of-pair score, can be improved. Iterative methods can be deterministic or stochastic, depending on the strategy used to improve the alignment. Traditional stochastic iterative methods include simulated annealing and genetic algorithms. Bioinformatics packages that use iterative algorithms are IterAlign, PRRP, DIALIGN, MUSCLE (Gotoh, 1996; Morgenstern et al., 1998; Berger and Munson, 1991).

The approach described in Barton and Sternberg (1987) is one of the first deterministic iterative algorithms. It uses information provided by a *profile*, which is a matrix that stores the relative frequency that each character appears in each column of an MSA. The following table shows an example of a profile (right) for an MSA of 4 sequences (left).

		1	2	3	4	...
S1: AABDB--BCC-CEAA	A	1.00	0.50	0.00	0.00	
S2: AA-DABBBCCCCE-A	B	0.00	0.25	0.75	0.00	
S3: A-BD--BBCCC-EAA	C	0.00	0.00	0.00	0.00	
S4: ABBDABBBCC--E-A	D	0.00	0.00	0.00	1.00	
	E	0.00	0.00	0.00	0.00	

The algorithm works as follows:

- i) Find two sequences with largest score and align them using standard DP (see Section 2.3).
- ii) Update the profile matrix by merging the two selected sequence in the previous step. Find a sequence that largest score to a profile of the alignment of the first two sequences and align it to the first two by profile-sequence alignment. Repeat until all sequences are included.
- iii) Remove randomly one sequence and create a profile of other sequences. Realign that sequence to profile alignment. Repeat for all sequences.
- iv) Repeat step 3 until either the score converges or a fixed number of iterations is reached.

Another iterative algorithm, DNR, was proposed in [Gotoh \(1996\)](#). It is a double nested iterative algorithm with randomization, which optimizes the weighted sum-of-pairs score function with the affine gap model. It starts with a preliminary MSA, which may be obtained by any more straightforward method. This MSA is the “seed” for all the calculations. Then, a set of weights is calculated by a phylogenetic tree.⁵ The algorithm consists of two nested loops. The first loop contains a randomized iterative method. After the convergence, a new phylogenetic tree and weights between pairs of sequences are calculated, and the second loop starts. In this way, the double nested iterations are continued until the total weighted sum-of-pairs score converges.

2.4.3 Other MSA Score Functions

Several other methods for assessing the quality of an MSA have been proposed in the literature, some of them being very distinct from those mentioned in the previous sections. For instance, some of them, such as BAliScore ([Thompson et al., 2005](#)) takes into account an existing reference MSA. In the following, we briefly mention some of them, with particular emphasis on BAliScore, Entropy ([Soto and Becerra, 2014](#)), MetAl [Blackburne and Whelan \(2012\)](#) and STRIKE ([Kemena et al., 2011](#)).

BAliScore This computes the accuracy of an MSA with respect to an existent reference MSA. It is provided together with the datasets in the benchmark database BALiBASE ([Thompson et al., 2005](#)) that consists of a large set of reference MSAs of known sequences. To compare the accuracy of an MSA with respect to the reference MSA in the benchmark, BAliScore computes two ratios: correctly aligned residue pairs (SP) and correctly aligned columns (TC) ([Thompson et al., 1999](#)). The score SP is the percentage of correctly aligned pairs of residues in the MSA. Consider an alignment of m sequences consisting of ℓ columns (see Definition 2.4.1 and notation therein). Let the j -th column in the MSA be denoted by $(a'_{1j}, a'_{2j}, \dots, a'_{mj})$, $j \in \{1, \dots, \ell\}$, where ℓ is the length of the MSA. For each pair of residues a'_{ij} and a'_{kj} , $i, k \in \{1, \dots, m\}$, let $p_{ikj} = 1$ if the residues a'_{ij} and a'_{kj} are aligned with each other in the reference MSA, and $p_{ikj} = 0$ otherwise. The S_j score for the j -th column is defined as follows:

⁵For reconstructing the guide tree, the UPGMA method is used ([Michener and Sokal, 1957](#)), which is constructed from the distance values between members in the initial alignment and is assigned to all pairs of sequences

$$S_j = \sum_{i=1}^{m-1} \sum_{k=i+1}^m p_{ikj}$$

The SP score for the MSA is then computed as follows:

$$\text{SP} = \frac{\sum_{j=1}^{\ell} S_j}{\ell^r \sum_{j=1}^{\ell} S_j^r}$$

where ℓ^r is the number of columns in the reference MSA and S_j^r is the score of the j -th column in the reference MSA.

The column score TC is the percentage of correctly aligned columns in the MSA. Let C_j be the score for the j -th column in the alignment, where $C_j = 1$ if all the residues in this column are aligned as in the reference MSA, otherwise $C_j = 0$, $j \in \{1, \dots, \ell\}$. The score TC for an MSA is then given as follows:

$$\text{TC} = \frac{\sum_{j=1}^{\ell} C_j}{\ell}$$

These ratios can be computed by the program BALiScore, which is publicly available at <http://www.lbgi.fr/balibase>. This program calculates the SP and TC only for special regions that are called *core blocks*. Core blocks correspond to the regions of the alignment that are *reliably* aligned. These regions are defined by some method based on a combination of secondary structure and sequence conservation (Thompson et al., 2005).

Entropy It measures the variability of an MSA by defining the frequencies of the occurrence of each character in each column. The total entropy for an MSA is the sum of the entropies of its columns (Soto and Becerra, 2014). Let ϕ be an MSA over an alphabet Σ' with ℓ columns (see Definition 2.4.1, page 25). Let $P_j(x)$ be the frequency of the occurrence of each character $x \in \Sigma'$ in each column j of the MSA, $j \in \{1, \dots, \ell\}$, where ℓ is the length of the MSA. Then, the entropy value $P_j(\phi)$ on each column j of the MSA ϕ is defined as follows:

$$P_j(\phi) = - \sum_{x \in \Sigma'} P_j(x) \log_2 P_j(x)$$

The entropy $E(\phi)$ of the MSA ϕ is the sum of entropies of its columns:

$$E(\phi) = \sum_{j=1}^{\ell} P_j(\phi)$$

Usually good alignments are considered to be those that minimize their total entropy.

Similarity *Similarity* can have a different meaning in bioinformatics but in this context is a measure that is defined by Kaya et al. (2014) in their method. It is a measure of similarity among all sequences. They first generate a position weight matrix from the alignment. A basic position weight matrix is created by counting the occurrences of each residue in each column and dividing by the number of sequences. Table 2.1 shows an example of an alignment and its corresponding position weight matrix. Then, the dominance value dv of the dominant residue in each column is calculated as follows:

$$dv(i) = \max\{f(a, i)\}$$

where $f(a, i)$ is the score of each residue a in column i , $i = \{1, \dots, \ell\}$. In the position weight matrix regardless of the existence of an indel, $dv(i)$ is the dominance value of the dominant residue on column i , and ℓ is the alignment length. For the alignment in Table 2.1, the dominant residue for column 1 is P and its dominance value is 0.75. The dominance values of the other columns are 1, 0.75, 1 and 1, respectively.

At last, the similarity objective function of alignment ϕ is defined as the average of the dominance values of all columns in the position weight matrix.

$$Similarity(\phi) = \frac{\sum_{i=1}^{\ell} dv(i)}{\ell}$$

For the above example, the similarity value, is computed as:

$$\left(\frac{0.75 + 1 + 0.75 + 1 + 1}{5} \right) = 0.9$$

Support Kaya et al. (2014) used this measure in their algorithm. The meaning of support is the same as that in the data mining field. Sometimes, if in the alignment we remove a sequence, the alignment score may increase. In other words, we can find alignment with a better score if a sequence that corrupts the alignment quality is removed. Therefore, support is the number of sequences present in alignment with a higher score. The greater the support value, the stronger is the alignment covered by most of the sequences in the dataset.

Table 2.1: An alignment with length 5 and its position weight matrix.

	1	2	3	4	5
F	-	F	F	D	
P	D	F	-	D	
P	-	F	F	D	
P	D	D	F	D	

 \implies

	1	2	3	4	5
F	0.25	0	0.75	1	0
P	0.75	0	0	0	0
D	0	1	0.25	0	1

MetAL. It is a scoring scheme that is obtained from four metrics to compare different MSAs. This score is based on: (i) a simple correction to the sum-of-pairs score; (ii) raw gap information; (iii) positions of gaps occurring in a sequence; and (iv) positions of indel events occurring both in a sequence and on its phylogenetic tree. The more detailed information regarding these metrics can be found in [Blackburne and Whelan \(2012\)](#). MetAl is available at <http://kumiho.smith.man.ac.uk/whelan/software/metal>.

STRIKE It is a scoring scheme to compare two alternative MSA from the same sequences by knowing the structural information of at least one the sequences ([Kemena et al., 2011](#)). STRIKE is available at <http://www.tcoffee.org>.

2.4.4 Benchmarks

Currently, many of data sets and techniques have been designed to standardize the comparison of sequence alignment results. Examples are OXBench ([Raghava et al., 2003](#)) HOMSTRAD ([de Bakker et al., 2001](#)) or Prefab ([Edgar, 2004](#)). In the thesis, for the purpose of the experimental analysis, we have chosen sequences obtained from the benchmark database BALiBASE 3.0 ([Thompson et al., 2005](#)), which is the most recent and used in the literature. The database contains high quality, manually constructed multiple sequence alignments together with detailed annotations. The alignments are all based on three-dimensional structural superposition.

BALiBASE is divided into five hierarchical reference data sets. The dataset Reference 1 consists of equidistant family sequences with two subgroups: RV11 and RV12. RV11 contains 38 datasets with less than 20% residue identity between groups and RV12 contains 44 data sets with residue identity between 20% and 40%. Reference 2 (RV20) contains 41 alignments comprising family sequences with more than 40% similarity and a highly divergent orphan sequence. Reference 3, with 30 datasets, contains subfamilies such that the sequences

within a given subfamily share more than 40% identity, but any two sequences from different subfamilies share less than 20% identity. The reference set RV40, with 49 datasets, contains sequences that are composed of groups with N/C-terminal extensions. The reference set RV50 contains 16 datasets with large internal insertions. We tested our approaches on all the instances from the sets RV11 and RV20, composed of sequences with different sizes and varying percentage identities.

2.5 Summary and Discussion

This chapter reviews the concept of the genomic data in Bioinformatics, sequence similarity, and sequence alignment. Then, it introduces the basic notation of scoring functions for pairwise and multiple sequence alignment and introduces dynamic programming approaches to solve the pairwise alignment based on global and local similarity. In addition, it describes the exact multiple sequence alignment algorithms in terms of sum-of-pairs score, followed by heuristic strategies that try to solve the multiple sequence alignment in a short amount of time.

Noteworthy, dynamic programming can be applied to most of the sequence alignment problems, at least, if two sequences are considered. The fact that the problem can be decomposed on subproblems based on the prefixes of the sequences gives a natural justification to the proper use of this paradigm. Interestingly enough, the heuristic methods reviewed in this chapter are far from being standard in the optimization field. Although the progressive approaches could resemble greedy algorithms that are guided by the guide tree, and iterative methods perform some kind of neighborhood search on the space of sequences, they do not fall into the usual classification of *metaheuristics*, such as genetic algorithms, tabu search, simulated annealing and so forth. In fact, the current optimization methods used in Bioinformatics are rather complicated and incorporate a wide knowledge from the practitioner to solve the problem at hand. Although they have been widely accepted by the Bioinformatics community, it is not yet clear whether better results could be achieved by following principled guidelines for the design of heuristic approaches to this problem, such as it has been observed for many other combinatorial optimization problems (Hoos and Stützle, 2004).

This chapter also shows that there has been a lot of effort in defining appropriate score functions. However, it is known that there is no “right” score function that assesses the true homology between the sequences (Morrison, 2015). One problem with the current approaches is that the usual score functions take into account a linear combination of several

components. Unfortunately, there is considerable disagreement about how to weight the several components (Gusfield et al., 1994), and with different weight combinations, different alignments can be achieved. Note, in addition, that the increase in the number of matches is followed by an increase of indels and/or gaps, which shows that the components are conflicting with each other.

A possible way of overcoming the problem is to consider a multiobjective formulation of this problem, which does not assume any definition of weights a priori. The alignment is then optimal with respect to this formulation if none of the components can be improved without degrading some of the other component values. The following chapter introduces notation and general algorithms to solve multiobjective optimization problems, which gives the basis for the design of algorithms to solve the multiobjective version of the sequence alignment problem.

Chapter 3

Multiobjective Optimization

3.1 Introduction

Many problems in bioinformatics and computational biology can be formulated as optimization problems (Handl et al., 2007). Multiobjective formulations are realistic models for many complex optimization problems. In most real-life problems, objectives conflict very often with each other, and optimizing with respect to a single objective can result in a poor solution for the remaining objectives. A reasonable approach to a multiobjective problem is to find a set of solutions, each of which cannot be improved in one objective without deteriorating at least one of the others.

This chapter introduces the basic concepts of multiobjective optimization. This class of problems can have different meanings according to different notions of optimality. Therefore, we need to introduce some of these notions but with particular emphasis on the notion of Pareto Optimality, which will be used throughout this thesis. Some of the existing solution approaches to solve multiobjective problems in terms of Pareto optimality are reviewed and discussed, from exact algorithms, which provide the set of optimal solutions with respect to this notion of optimality, to some relevant heuristic methods, which return an approximation of bounded size. In the end, performance assessment methods for heuristic approaches are explained in detail.

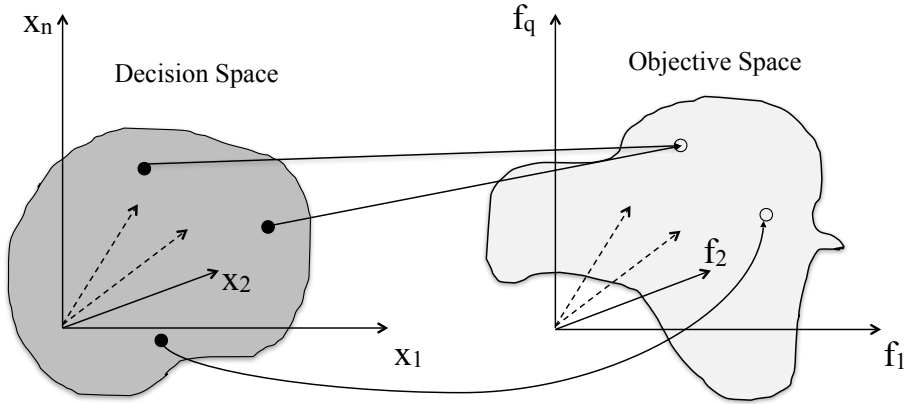


Figure 3.1: Mapping of the n -dimensional space to the q -dimensional objective space

3.2 Notation and Definitions

A multiobjective optimization problem (MOP) is the problem of finding values for a set of decision variables that optimize a vector objective function given a set of constraints. The components of this function form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term “optimize” means finding a solution that would be as good as possible to all objectives. A general MOP can mathematically be defined as follows:

$$\begin{aligned} \text{“maximize” } z = f(x) &= (f_1(x), f_2(x), \dots, f_q(x)) \\ \text{s.t. } x &\in X \end{aligned} \tag{3.1}$$

where x is an n -dimensional decision vector, or solution, and X is the set of all feasible solutions.

Note that if some objective function is to be maximized, it is equivalent to maximize its negative. Without loss of generality, we assume *maximization* of all objectives. The objective function f maps X into \mathbb{R}^q , where $q \geq 2$ is the number of objectives and the vector $z = f(x)$ is an objective vector or point; see an illustration of these concepts in Fig. 3.1.

The term “maximize” appears above in quotation marks because its meaning is not yet defined. Alternative formulations of a MOP exist, such as Pareto optimality, lexicographic optimality, maximization of the minimum of all the objectives (maxmin), and maximization of a scalarized combination of the objectives (Ehrgott, 2005). Pareto optimality is the notion

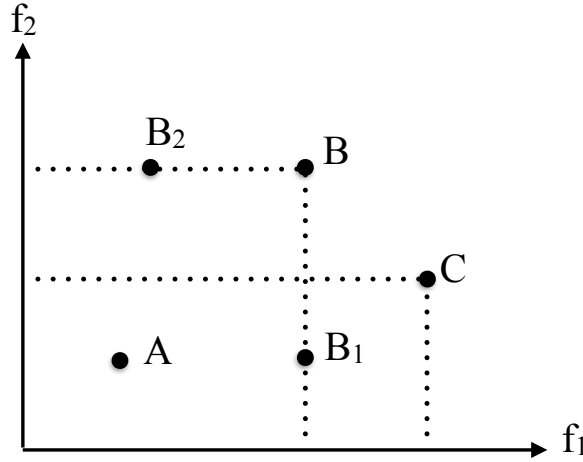


Figure 3.2: A schematic illustration of the “dominance relation”, assuming two maximizing objectives

of optimality that requires less information about the preferences of a decision maker. In the following, we specify the notion of “maximize” that is considered in this work.

3.2.1 Pareto Optimality

The main goal of solving MOPs in terms of Pareto optimality is to find solutions that are not worse than any other solution and strictly better in at least one of the objectives. To understand how the corresponding objective vectors can be ordered, the following binary relations in \mathbb{R}^q are introduced (Ehrgott, 2005). Let u and v be vectors in \mathbb{R}^q .

- $u \succeq v \iff u_i \geq v_i, i = 1, \dots, q;$
- $u \succ v \iff u \neq v \text{ and } u_i \geq v_i, i = 1, \dots, q;$
- $u > v \iff u_i > v_i, i = 1, \dots, q.$

In the context of optimization, we denote the relation between objective function vectors of two feasible solutions x and x' as follows:

- if $f(x) \succeq f(x')$, then $f(x)$ weakly dominates $f(x')$.
- if $f(x) \succ f(x')$, then $f(x)$ dominates $f(x')$;

- if $f(x) > f(x')$, then $f(x)$ strictly dominates $f(x')$;

Figure 3.2 illustrates the concept of dominance in the biobjective case, assuming maximization. In this example, the point B strictly dominates point A . Point C strictly dominates points A and B_1 . Point B_1 and B_2 are mutually non-dominated, but they are (weakly) dominated by point B .

For simplification purposes, we shall use the same notation among solutions when the above relations hold between their objective function vectors. Since the notion of optimal solution clearly differs from the single-objective counterpart, we define the notion of optimality in multiobjective optimization:

Definition 3.2.1. (*Pareto optimal solution*) A feasible solution $x^* \in X$ is Pareto optimal if there exists no $x \in X$ such that $f(x) \succ f(x^*)$.

Definition 3.2.2. (*Pareto optimal set*) $X' \subseteq X$ is the Pareto optimal set if and only if it contains only and all Pareto optimal solutions.

The image of a Pareto optimal solution in the objective space is a *non-dominated point* and the set of all non-dominated points is called *non-dominated point set*. Solving a MOP in terms of Pareto optimality would correspond to find the *Pareto optimal set*. However, rather than finding all solutions in this set, the non-dominated point set may be preferable, since, in practice, the decision maker takes his decision by inspecting the objective space.

In many real-life applications of multiobjective optimization, only approximations to the non-dominated point set are required since there might be many or infinite numbers of non-dominated points. In the following, we give a definition of the outcome of these approaches.

Definition 3.2.3. (*Approximation set*) Any approximation to the non-dominated point set is called *approximation set*. If A is an approximation set then the elements of A are mutually non-dominated.

In the following, we introduce two different notions of optimality in multiobjective optimization that are related to Pareto optimality.

3.2.2 Lexicographic Optimality

In many cases, the decision maker is able to rank the objectives according to their relevance. This notion of optimality considers a ranking of the objectives according to lexicographic ordering: given two vectors u and v in \mathbb{R}^q , $u >_{lex} v$ if there exists a $j \in \{1, \dots, q-1\}$ such that $u_i = v_i$, $\forall i = 1, \dots, j$ and $u_{j+1} > v_{j+1}$.

Definition 3.2.4. (*Lexicographic optimal solution*) A solution $x^* \in X$ is lexicographic optimal solution if and only if there exists no $x \in X$ such that $f(x) >_{lex} f(x^*)$.

This definition implies that the objectives are ordered according to decreasing importance (Ehrgott, 2005). Usually, methods for obtaining an optimal lexicographic solution are based on solving each objective sequentially by decreasing the order of priority and using the optimal solutions of higher priority objectives as constraint values. An essential property of this notion of optimality is that an optimal lexicographic solution is also a Pareto optimal solution for the same multiobjective problem (Ehrgott, 2005).

3.2.3 Scalarized Optimality

When the decision maker is able to weight the importance of each objective, it is possible to solve a MOP in terms of scalarized optimality. The objective function vector is scalarized with a weight vector $\lambda = (\lambda_1, \dots, \lambda_q) \in \mathbb{R}_{\geq 0}^q \setminus \{\mathbf{0}\}$. The weight vector λ is usually normalized such that $\sum_{i=1}^q \lambda_i = 1$. The scalarized objective function is then denoted by $f_\lambda(x)$ (Ehrgott, 2005).

Definition 3.2.5. (*Scalarized optimal solution*) A solution $x^* \in X$ is a scalarized optimal solution if and only if there is no $x \in X$ such that $f_\lambda(x) > f_\lambda(x^*)$ with respect to a given λ .

In the case of different ranges of values between objectives, normalization by range equalization factors must be considered.

The scalarization of the objective function vector is usually based on the family of weighted L_p -metrics as follows:

$$f_\lambda(x) = \left(\sum_{i=1}^q (\lambda_i |f_i(x) - z_i|^p) \right)^{1/p} \quad (3.2)$$

where $x \in X$, $p > 0$, $z^* = (z_1, \dots, z_q)$ is the ideal vector defined as $z_i = \max_{x \in X} f_i(x)$, for $x \in X$. Usually, $p = 1$ or $p = \infty$ are often used. When $p = 1$, Eq. 3.2 becomes:

$$f_\lambda(x) = \sum_{i=1}^q \lambda_i |f_i(x) - z_i|. \quad (3.3)$$

which is equivalent to the well-known weighted sum formulation:

$$f_\lambda(x) = \sum_{i=1}^q \lambda_i f_i(x). \quad (3.4)$$

For $q = \infty$ the above formula gives:

$$f_\lambda(x) = \max_{i=1,\dots,q} (\lambda_i |f_i(x) - z_i|). \quad (3.5)$$

The following relation between weighted sum scalarization and Pareto optimal solutions is explored in this thesis: An optimal solution for the weighted sum is also a Pareto optimal solution for the same problem (Ehrgott, 2005). However, the contrary is not true in general.

3.3 Solution Methods for Multiobjective Optimization

This section reviews some of the existing solution methods to solve MOPs in terms of Pareto optimality, ranging from exact algorithms that find the Pareto optimal set to heuristic approaches that return an approximation set.

3.3.1 Exact Methods

Exact methods can solve optimization problems to optimality, and they can range from very general techniques, such as branch-and-bound, to problem-dependent algorithms, such as Dijkstra's algorithms for the shortest path problem. However, for most MOPs, exact algorithms can take exponential time to terminate. There could be a Pareto optimal set of intractable size, which makes the use of exact algorithms even less appealing. In addition, many MOPs whose single objective version are solvable in polynomial time become NP-hard when one more objective is added (Ehrgott, 2005). This means that extensions of well-known polynomial time exact algorithms cannot efficiently solve the corresponding MOP versions. This is the case for the shortest path problem, the minimum spanning tree problem, etc.

Despite these negative results, many exact approaches have been discussed in the literature (Ehrgott, 2005). In fact, some result obtained in Mote et al. (1991) indicate that some highly specific exact algorithms can achieve the Pareto optimal set in a reasonable amount of time, depending on some instance feature.

In the following, several general techniques are discussed. They are based on consecutive runs of an underlying algorithm that is able to solve the single objective counterpart of the MOP. Two main approaches exist, those based on solving several scalarized versions of the original MOP and others based on the optimization of one objective while the others are being used as constraints.

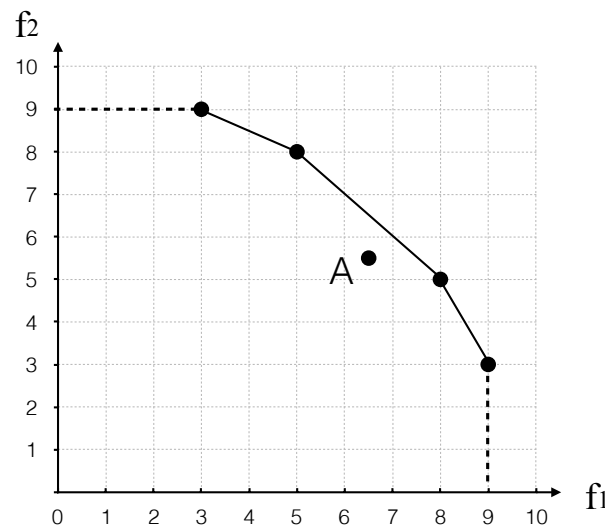


Figure 3.3: Illustration of the weighted-sum method

Methods Based on the Weighted-sum Scalarization

A well-known way of solving a MOP is to scalarize the objective function vector for different weight vectors as a weighted-sum (see Section 3.2.3). A given weight vector λ can be interpreted as a given search direction in the objective space. Figure 3.3 illustrates an example with 5 non-dominated points. For $\lambda = (0.6, 0.4)$, the point $(8, 5)$ is the maximum for the scalarization with respect to λ . It is important to note that it is not possible to construct the whole Pareto optimal set by applying this method. In discrete optimization problems, there exist Pareto optimal solutions that are not optimal for the weighted-sum scalarization (Ehrgott, 2005). Those solutions are called *non-supported*. In Figure 3.3, point $A = (6.5, 5.5)$ is a non-supported point that is not optimal for any value of λ .

Definition 3.3.1. (*Supported (non-supported) solutions*) A Pareto optimal solution that is (not) optimal for a (any) weighted sum with non-negative weight vector is a supported (non-supported) solution.

Many methods based on solving several scalarized versions of the original MOP return a set of solutions, each of which is obtained by solving a certain scalarization of the objective function vector by means of Eq. (3.4) (weighted sum). Assuming that the non-dominated score set has a convex shape in the objective space toward the improvement of all objectives, all its elements can be achieved by the appropriate weight vectors. However, convexity

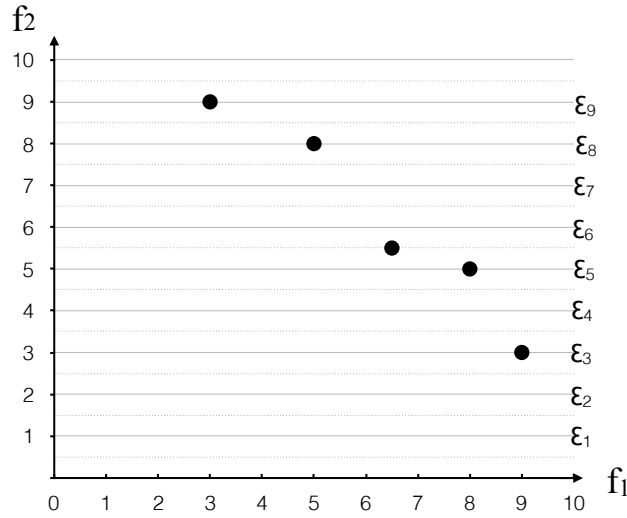


Figure 3.4: A schematic illustration of the ϵ -constraint method, assuming biobjective optimization, maximizing f_1 subject to the constraint f_2

should not be expected to hold in the general case, since non-convex shapes can also occur for MOPs of discrete nature (Ehrgott, 2005). Therefore, this approach is not suitable for many problems.

Nevertheless, weighted-sum approaches are still applied to MOPs where the convexity of the non-dominated score set in the objective space does not hold, for example, for generating supported solutions that are used as starting solutions for another solution method. An alternative to the weighted-sum formulation is to consider Eq. (3.5), where the distance from the ideal vector is to be minimized. If the underlying algorithm is applied to several weights, some non-supported solutions can indeed be obtained, but it is not possible to distinguish between Pareto optimal solutions and the solutions that are weakly dominated by them. Moreover, these formulations imply the need to add further constraints, making the problem harder to solve.

ϵ -constraint Method

The ϵ -constraint method (Haimes et al., 1971) is another scalarized approach. It converts the MOP problem into a single objective problem with $q - 1$ constraints. The principle of ϵ -constraint method is based on optimizing one objective while others are used as constraints. The advantage of the ϵ -constraint method is that it can obtain all the non-dominated set, even

if convexity in the objective space does not hold (Ehrgott, 2005). However, the disadvantage of this method is that the number of single objective problems to solve may be very large since it depends on the range of objective values for each objective function.

The ϵ -constraint method consists of solving, iteratively, a single objective problem in $q-1$ constraints, each of which is defined with respect to a given right-hand side (ϵ). The right-hand of each constraint is varied iteratively during the run of this method. The formulation of ϵ -constraint in a biobjective problem is as follows:

$$\begin{aligned} \max \quad & f_1(x) \\ \text{s.t.} \quad & f_2(x) \geq \epsilon \\ & x \in X \end{aligned}$$

where $f_1(x)$ is the objective to be maximized while the constraint is defined with respect to $f_2(x)$. The solutions that can be found for the underlying MOP largely depend on the selection of the values of ϵ . In particular, they must be chosen such that they lie between the minimum and maximum value of each objective function. Moreover, if the gap between two consecutive ϵ values is too large, non-dominated points may be missed. Figure 3.4 shows an illustration of the ϵ -constraint method in a biobjective problem where the constraint is defined with respect to objective f_2 . In this case, if the value of ϵ is considered to varying one by one, then the point (6.5,5.5) will be missed.

3.3.2 Heuristic Methods

Despite the hardness of many MOPs, these problems still need to be solved. Therefore, high-quality approximations are usually required in many practical applications. One class of these methods that provide approximations are heuristic algorithms. In the following, we review some of the heuristic methods that have been successfully applied to MOPs.

Evolutionary Algorithms

Evolutionary algorithms (EAs) are inspired by natural evolution and the Darwinian concept of “Survival of the Fittest” (Coello et al., 2007). Several EAs have emerged as flexible and robust *metaheuristic* methods for solving optimization problems. In this section, we explain the basic concepts and terminology in EA and provide a generic structure for these approaches. Next, we review existing multiobjective evolutionary algorithms, with a particular focus on MOGA (Fonseca and Fleming, 1993) and NSGA-II (Deb et al., 2002).

Basic Concepts of EA In general, an EA maintains a set of candidate solutions that goes through a selection process and is altered by genetic operators, usually *recombination* (crossover) and/or *mutation*. In analogy to the natural evolution, a candidate solution is an *individual* and a set of candidate solutions is a *population*. Each individual may represent a possible solution to the problem, although it can also represent an infeasible solution to over-constrained problems.

In the selection process, which is usually probabilistic, low-quality individuals are removed from the population, while high-quality individuals are reproduced. The goal is to increase the average quality of the population. The quality of an individual in a single-objective optimization problem is represented by a scalar value, the *fitness value*, which, in most cases, corresponds to the objective value of the solution it represents. Crossover and mutation aim at generating new solutions by changing the existing ones. The crossover operator takes a certain number of individuals in the population (parents) and creates a certain number of new individuals (children) by recombining the former. To imitate the stochastic nature of evolution, a crossover probability is associated with this operator. The mutation operator modifies individuals according to a given mutation rate. Both crossover and mutation operators work on individuals, i.e., in the decision space.

Of particular importance is how the solution is “computationally” represented by an individual since this affects the choice of the genetic operator. In the following explanation, we assume optimization problems with n integer variables. Therefore, an individual is represented by a vector of n elements and stores, at each position i , the value of the i -th variable, $i = 1, \dots, n$.

Generic Structure of an EA Algorithm 1 shows the basic structure of an EA. Each loop iteration is called a *generation*. The population P at a certain generation t is represented by P_t , the offspring population is denoted by Q_t and N is the population size. First, an initial population is created. Then, a loop to generate offspring consisting of the steps crossover, and/or mutation, evaluation (fitness assignment) and selection is executed until a termination criterion is satisfied. In the end, the best individual(s) in the final population or found during the entire search process is the outcome of the EA. In the following, these terminologies are explained more in details.

- *Initialize population*: This is a crucial task in EAs since it can affect the convergence speed and also the quality of the final solutions. If no information about the problem

Algorithm 1 The general scheme of Evolutionary Algorithms

procedure GENERAL EVOLUTIONARY ALGORITHM:

 $t = 0$
Initialize population with candidate individuals

 Generate N individuals to form the first population P_0 .

 Evaluate the fitness of individuals in P_0 .

while *termination criterion* is not satisfied **do**

 Generate offspring population Q_t :

 Crossover: Choose parents individuals x and y from P_t and use a crossover operator to generate offspring and add them to Q_t .

 Mutation: Mutate some individual $x \in Q_t$ with a predefined mutation rate.

 Fitness assignment: Evaluate and assign a fitness value to each individual $x \in Q_t$ based on its quality.

 Selection: Select N individuals from Q_t based on their fitness and assign them to P_{t+1} .

 $t = t + 1$
return the current population P_t

is available, then random initialization is the most commonly used method to generate the initial population.

- *Crossover and/or Mutation*: The most used *crossover* method is *single point crossover*. Given two parents A and B , and a given value i randomly chosen between 1 and n , the single point crossover generates a new individual by merging the first i elements from A with the last $n - i$ elements from B . A similar procedure is performed to generate a second individual by exchanging A with B . Another type of crossover is the *double-point crossover*, where two merging points are considered. Figure 3.5 illustrates the two operators. In principle, crossover operation leads the population to converge by making the individuals in the population alike. In mutation, small parts of an individual are altered or replaced with a new one to create a new individual. Figure 3.6 shows an example. It is expected that mutation introduces diversity into the population and assists the search process to escape from local optima.
- *Fitness Assignment*: Fitness assignment consists of assigning a measure of quality, the *fitness value*, to an individual. Usually, in optimization, the fitness value of an

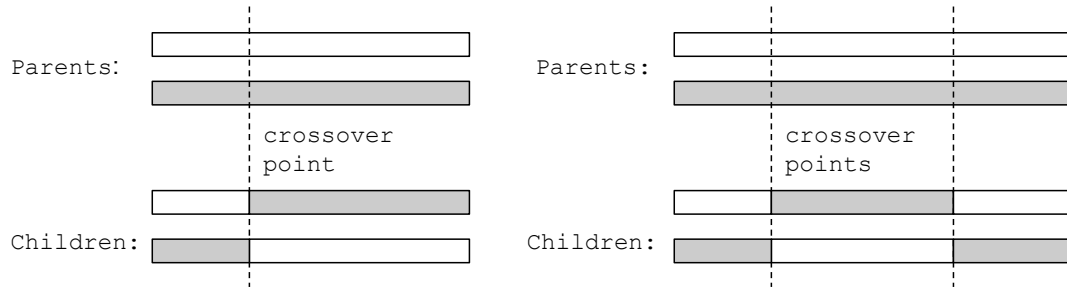


Figure 3.5: Illustration of single point (left) and double point (right) crossover operator

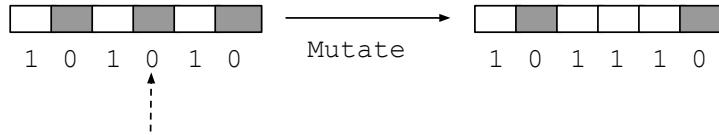


Figure 3.6: Illustration of mutation operator

individual is the objective function value of the solution it represents.

- *Selection*: There are different selection procedures in EAs depending on how the fitness values are used. Stochastic universal sampling, proportional sampling, and tournament selection are the most popular ones. In the following, there is a brief explanation of how each of them works:
 - *Tournament selection*. In this selection process, some individuals are randomly chosen from the population and several “tournaments” runs among them. The winner of each tournament (the one with the best fitness) is selected for crossover.
 - *Fitness proportionate selection (FPS)*. An individual is selected in a probabilistic manner. Let f_j be the fitness of the j -th individual in the population. Then, the probability of being selected is given by $f_j / \sum_{i=1}^N f_i$, where N is the number of individuals in the population.
 - *Stochastic universal sampling (SUS)*. This is similar to FPS, but with no selection bias. SUS uses a single random value to sample all of the individuals by choosing them at evenly spaced intervals. This gives weaker members of the population (according to their fitness) a chance to be chosen and thus reducing the unfair nature of FPS selection methods.

- *Elitist selection.* In this selection process, the best solutions according to their fitness values always remain in the population.
- *Termination criterion:* To stop the iteration process, a different type of termination criterion may be used such as a predefined maximum number of generations, stagnation in the population or existence of an individual with sufficient quality.

Multiobjective Evolutionary Algorithms

There is a wide belief that EAs can be applied successfully to MOP because they can handle multiple solutions simultaneously. In fact, EAs can be easily modified in order to return multiple non-dominated solutions in a single simulation run. Such EAs are known as *Multiobjective Evolutionary Algorithms* (MEAs). However, since they are heuristic methods, MEAs cannot generate the entire Pareto optimal set in general. Moreover, they may fail to find a single Pareto optimal solution. For this reason, they return approximation sets. Clearly, there is the need to define the *quality* of an approximation set, which should characterize the performance of the MEA that produced it. Therefore, the goal of an MEA has been to produce approximation sets that minimize the distance from the optimal non-dominated point set as well as to maximize the distribution and spread of its elements (Zitzler and Thiele, 1999).

The development of an MEA for a given MOP should address the following issues:

- How to accomplish fitness assignment and selection, respectively, in order to guide the search towards the non-dominated point set.
- How to maintain a diverse population in order to avoid early convergence and achieve a well-distributed and well-spread approximation set.

To deal with these issues different techniques have been applied, which are explained in the following sections.

- *Fitness assignment and selection* In contrast to single-objective optimization, where objective function and fitness function are the same, both fitness assignment and selection must allow for several objectives values. In general, there are two main types of MEAs:
 - Algorithms that are based on the classical scalarization techniques (e.g., approaches that use a weighted-sum scalarization);

- Algorithms that use dominance relations (e.g., rank the population based on dominance relation).
- *Population diversity.* To have a diverse population, several techniques have been developed. The most frequent ones are as follows:
 - *Fitness Sharing.* In this technique, the subset of individuals that are close to each other (niche) shares their fitness values. This corresponds to a decrease in their fitness value, which gives them less chance of surviving in the selection process (Fonseca and Fleming, 1993).
 - *Crowding distance.* Crowding distance approaches aim to obtain a uniform spread of solutions without fitness sharing. They compute the population density around a solution without requiring a user-defined parameter, and less crowded solutions are emphasized in the selection process (Deb et al., 2002).
- *Elitism.* This policy always selects the best individual of a population, according to dominance relation or to other quality measures, in order to prevent losing it due to sampling effects or genetic operators. This strategy can be extended to $k > 1$ best solutions in the population.

The first MEA was proposed by Schaffer (1985), called Vector Evaluated Genetic Algorithms (or VEGA). Afterwards, several MEAs were developed such as Multiobjective Genetic Algorithm (MOGA) (Fonseca and Fleming, 1993), Niche Pareto Genetic Algorithm (Horn et al., 1994), Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb, 1994), Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1999), Pareto-Archived Evolution Strategy (PAES) (Knowles and Corne, 2000) and Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb et al., 2000). In the following, we discuss two MEAs that use a selection process based on the dominance relation.

Multiobjective Genetic Algorithm (MOGA) This MEA was proposed by Fonseca and Fleming (1993), which employed the concept of niche along with rank-based fitness assignment, named *Pareto ranking*. Pareto ranking is a process for ranking individuals in a population based on the distances of their objective vectors from the non-dominated set obtained so far during the search process. Individuals that have an objective vector closer to the non-dominated set are better ranked. The closeness to the non-dominated set is

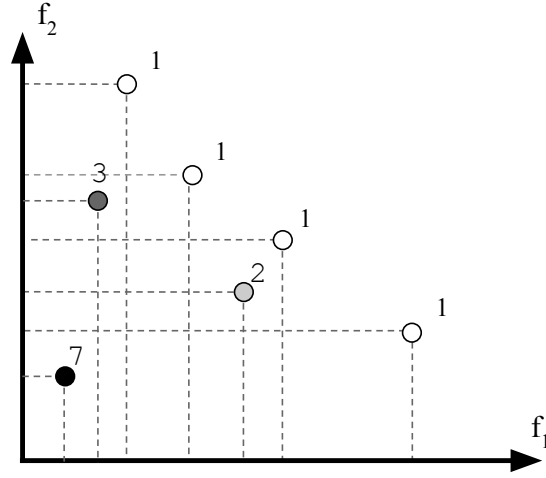


Figure 3.7: Illustration of Pareto ranking in MOGA

determined in terms of dominance relations. For simplification purpose, we use the same notation among individuals when the dominance relations (see Section 3.2.1) hold between their objective function vectors. Let r_i be a rank of each individual i . It is assigned as:

$$r_i = 1 + n_i \quad (3.6)$$

where n_i is the number of individuals that dominate individual i . Since n_i is zero for non-dominated individuals in the population, non-dominated individuals are assigned a rank of 1. Figure 3.7 illustrates a hypothetical population and the corresponding ranks of the individuals. The individuals whose associated objective vector are non-dominated have rank 1 while the worst individual has rank 7.

Based on the Pareto ranking, a fitness value is assigned to each individual so that individuals with better ranks have larger fitness values. If N is the population size, and $NR(r_i)$ is the number of individuals with rank r_i , the fitness value obtained by the following formula:

$$F'(i) = N - 0.5(NR(r_i) - 1) - \sum_{k=1}^{r_i-1} NR(k) \quad (3.7)$$

The fitness value varies from 1 to N depending on the rank of each individual and the number of individuals in each rank.

In order to impose diversity in the population, MOGA employs the sharing function approach for each individual i . More formally, the shared fitness $F(i)$ for each individual i in population \mathcal{P} is equal to its old fitness $F'(i)$ (Equation 3.7) divided by its *niche count*.

Algorithm 2 The fitness assignment algorithm in MOGA

procedure FITNESS ASSIGNMENT (\mathcal{P} , σ_{share})

for each $i \in \mathcal{P}$ **do**

$$r_i = 1 + n_i$$

 Sort population \mathcal{P} according to ranking r

for each $i \in \mathcal{P}$ **do**

 Calculate fitness values $F(i)$

return fitness values of \mathcal{P}

The niche count for each individual is equal to the sum of the sharing function (SH) values for all of the individuals with the same rank in the population:

$$F(i) = \frac{F'(i)}{NR(r_i) \sum_{j=1} SH(d(i, j))} \quad (3.8)$$

where the standard sharing function $SH(\cdot)$, as suggested by Goldberg et al. (1987) is defined as follows:

$$SH(d(i, j)) = \begin{cases} 1 - \left(\frac{d(i, j)}{\sigma_{share}} \right) & \text{if } d(i, j) < \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \quad (3.9)$$

where σ_{share} is the niche radius and $d(i, j)$ is a distance function. A normalized niche radius σ_{share} in the q -dimensional objective space and population of size N is calculated as follows (Fonseca and Fleming, 1993):

$$(1 + \sigma_{share})^q = N \times (\sigma_{share})^q \quad (3.10)$$

A normalized distance between any two individuals i and j of the same rank is calculated in the objective space as follows:

$$d(i, j) = \sqrt{\sum_{k=1}^q \left(\frac{f_{k,i} - f_{k,j}}{\max f_k - \min f_k} \right)^2} \quad (3.11)$$

where $f_{k,i}$ and $f_{k,j}$ are the k -th objective function for individuals i and j , respectively, and $\max f_k$ and $\min f_k$ are the maximum and minimum objective function values of the k -th objective function observed so far during the search, respectively.

The high-level pseudo-code of Algorithm 2 shows the steps of fitness assignment procedure in MOGA. First, it calculates the rank for each individual according to the dominance relation and, then, each individual is assigned a fitness value based on its rank in the population (Eq. (3.7)). To maintain diversity, it adjusts the fitness function according to sharing function (Eq. (3.8)).

Miller and Shaw (1996) proposed a dynamic niche sharing approach to increase the effectiveness of computing niche counts. Note that one of the disadvantages of MOGA in fitness sharing based on niche count is its computational effort. However, its benefits surpass the burden of extra computational effort in many applications.

Non-dominated Sorting Genetic Algorithm (NSGA-II) NSGA-II (Deb et al., 2000, 2002) is the second version of the NSGA (Srinivas and Deb, 1994), “Non-dominated Sorting Genetic Algorithm”, to solve multiobjective optimization problems. In the following, we explain the working principles of this algorithm with respect to the choice for sorting the individuals in the population as well as strategies for keeping diversity and elitism.

- *Fast non-dominated sorting.* NSGA-II ranks the individuals in the population into layers based on dominance relation. Any two individuals in the same layer are non-dominated. An individual belongs to a certain layer if it is not dominated by any individual of the inferior layer. The first layer is computed by finding the individuals that are not dominated by any other individuals in a population. The next layers are computed similarly by ignoring the individuals that belong to the previous layers. Figure 3.8 (left) shows an illustrative example of the ranking in NSGA-II. The first layer \mathcal{L}_1 is shown with white circles. By putting aside them, the second non-dominated layer \mathcal{L}_2 is shown with gray circles and the last individual create the last layer (black circle). Efficient algorithms for the ranking are discussed in Jensen (2003).
- *Diversity.* In order to obtain individuals uniformly distributed over the non-dominated set, NSGA-II uses the *crowding distance*. For a particular individual in the population, the algorithm estimates its density by calculating the average distance from other individuals around it on each objective. Figure 3.8 (right) illustrates the crowding distance calculation, where $dist(i)$ is the estimation of density of individual i inside its non-dominated layer, which is equal to the size of largest cuboid surrounding i , without considering any other individuals in the population (shown with a dashed box). Algorithm 3 shows the calculation of the crowding distance for each individual

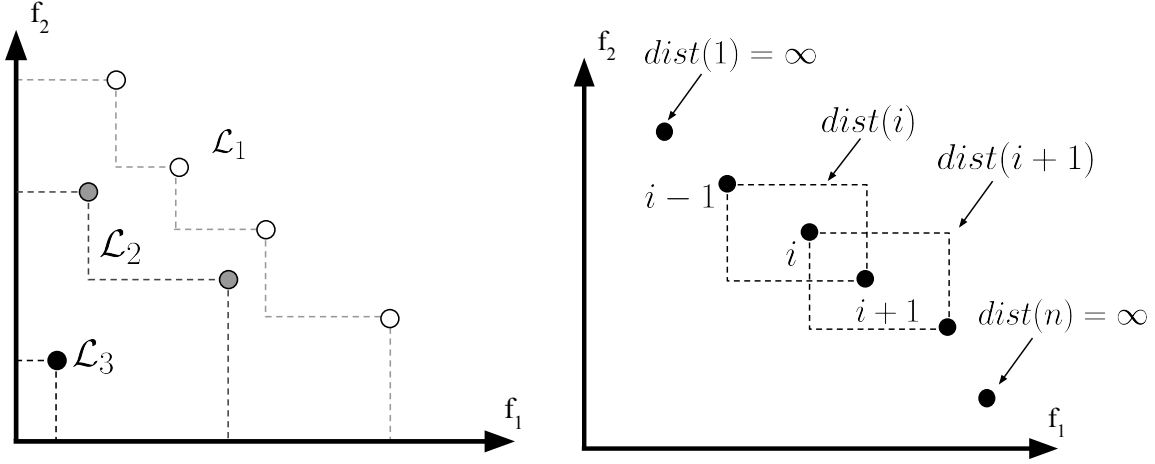


Figure 3.8: Illustration of ranking (left) and crowding distance (right) in NSGA-II

in a given layer. For each objective function k , $k = 1, \dots, q$, the algorithm sorts the individuals in \mathcal{L} in non-increasing order; $f_{k,i}^{\mathcal{L}}$ refers to the k -th objective function value of the i -th individual in layer \mathcal{L} . The main advantage of the crowding approach is that a measure of population density around a solution is calculated without requiring a sharing parameter (σ_{share}). In the NSGA-II, in the selection phase, this crowding distance measure is used as a tiebreaker. In order to compare solutions, NSGA-II introduces the *crowded comparison operator* \leq_n . Each individual i in the population has two attributes: the layer that belongs to ($rank(i)$) and the local crowding distance ($dist(i)$). Then, the crowded comparison operator is defined as follows:

$$j \leq_n i \quad \text{if } rank(i) < rank(j) \text{ or } (rank(i) = rank(j) \text{ and } dist(i) > dist(j))$$

If two individuals belong to different layers, the individual from the layer with the lowest index is selected; otherwise, if both of them belong to the same layer, then the individual with the highest crowding distance (that is located in a region with a lesser number of individuals) is selected.

- *Elitism.* Elitism in the context of single-objective GA means that the best solutions found so far during the search always survive in the next generation. In this respect, all non-dominated solutions discovered by a multiobjective GA are considered as elite solutions. However, implementation of elitism in multiobjective optimization is not as straightforward as in single objective optimization mainly due to a large number of possible elitist solutions. NSGA-II uses two strategies to implement elitism: it

Algorithm 3 The algorithm for calculating the crowding distance in NSGA-II

procedure CROWDING DISTANCE CALCULATION(\mathcal{L})

$n_{\mathcal{L}} = |\mathcal{L}|$

for each $i \in \mathcal{L}$ **do**

$dist(i) = 0$

for $k = 1$ **to** q **do**

Sort individuals in layer \mathcal{L} in non-increasing order of objective k

$dist(1) = dist(n_{\mathcal{L}}) = \infty$

for $i = 2$ **to** $n - 1$ **do**

$dist(i) = dist(i) + f_{k,i+1}^{\mathcal{L}} - f_{k,i-1}^{\mathcal{L}}$

maintains elitist solutions in the population, and stores elitist solutions in an external secondary list and reintroduces them into the population.

Algorithm 4 The main loop of NSGA-II algorithm

procedure MAIN ITERATION:

Generate layers $\mathcal{L}_1, \dots, \mathcal{L}_R$ from $P_t \cup Q_t$ with fast non-dominated sorting

$i = 1$

$P = P_{t+1} = \emptyset$

while $|P| < N$ **do**

$CrowdingDistanceCalculation(\mathcal{L}_i)$

$P = P \cup \mathcal{L}_i$

$i = i + 1$

Sort P in non-decreasing order of crowding distance

P_{t+1} contains the best N individuals from P

Generate Q_{t+1} from P_{t+1} by evolutionary operators

$t = t + 1$

In the following, we describe how NSGA-II works. Initially, a randomly generated population P_0 of N elements is created. The population is sorted into layers based on the fast non-dominated sorting step. Since the maximization of the objectives is considered, each individual is assigned with a fitness equal to the reciprocal of the non-dominated layer it belongs to (the best layer is 1). To create an offspring population Q_0 , binary tournament selection, crossover and mutation operators are applied. After the first generation is created,

the procedure will keep the N best individuals (elitism strategy). Algorithm 4 shows the main loop of NSGA-II at iteration t . Non-dominated layers $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_R$ are identified in the combined population $P_t \cup Q_t$. The next population P_{t+1} is filled starting with individuals from \mathcal{L}_1 , then \mathcal{L}_2 , and so on as follows. Let k be the index of a non-dominated layer \mathcal{L}_k such that $|\mathcal{L}_1 \cup \mathcal{L}_2 \cup \dots \cup \mathcal{L}_k| \leq N$ and $|\mathcal{L}_1 \cup \mathcal{L}_2 \cup \dots \cup \mathcal{L}_k \cup \mathcal{L}_{k+1}| > N$. First, all non-dominated individuals in the levels $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k$ are copied to P_{t+1} , and then the $N - |P_{t+1}|$ with the least crowded distance from \mathcal{L}_{k+1} are added to P_{t+1} . This approach ensures that all non-dominated individuals from \mathcal{L}_1 are included in the next population if $|\mathcal{L}_1| \leq N$, and, otherwise, the selection based on a crowding distance is used to keep diversity. Finally, population Q_{t+1} is obtained from P_{t+1} by the application of selection, crossover and mutation.

Algorithm 5 The general outline of an Local Search algorithm

procedure LOCAL SEARCH:

Determine initial *candidate solution*

Initiate *incumbent solution* with *candidate solution*

while termination criterion is not satisfied **do**

Perform *search step*

If necessary, update *incumbent solution*

return incumbent solution or report failure

Local Search

Local Search (LS) techniques vary from simple iterative improvement algorithms to general algorithm frameworks such as Simulated Annealing and Tabu Search, which can be adapted to a specific problem. In general, they are based on the idea of iteratively improving a candidate solution by means of small modifications. Most of them use randomized decisions in the search process such as generating an initial solution or determining the search step.

Algorithm 5 illustrates the working principle of LS algorithms. They need to keep track of the best solution (*incumbent solution*) found during the search process. Among the components underlying the LS algorithm, the *neighborhood* definition and the *step function* are important. The set of all solutions that are obtained by applying a given modification in the current solution is called the neighborhood, and it is very problem specific. It is often difficult to predict which of the various neighborhood definitions results in the best performance. The step function describes how to move between *candidate solutions*. *Search steps*

are usually defined by means of a procedure that draws a sample from the probability distribution determined by the underlying *step function*. The search process is typically guided by an evaluation function that is used to assess or rank candidate solutions heuristically.

There exist two basically different LS models for MOPs (Paquete and Stützle, 2007). The first approach consists of constructing the search step based on the dominance relation (see Section 3.2.1). The LS algorithms that use this approach follow the *component wise acceptance criterion* (CWAC) search model. The second approach is based on the parameterized scalarization of the objectives (see Section 3.2.3), known as *scalarized acceptance criterion* (SAC) search model. Most LS algorithms for MOPs return a set of non-dominated solutions, which are created during the search process. The best non-dominated solutions found during the run are maintained into an archive. During the search process, the algorithm needs to update the archive, and this update consists of adding new non-dominated solutions and removing the dominated ones. In the following, we discuss a particular LS approach that follows the CWAC model, which will be used in the context of this thesis.

Algorithm 6 The general scheme of Pareto Local Search (PLS)

procedure PLS: PARETO LOCAL SEARCH (initial solution: x_0)

$visited(x_0) = \text{false}$

$A = \{x_0\}$

repeat

 Choose randomly $x \in A$ s.t. $visited(x) = \text{false}$

for all $x' \in \text{Neighbors}(x)$ **do**

if $f(x'') \not\prec f(x')$, for all $x'' \in A$ **then**

$A = A \cup \{x'\}$

$A = A \setminus \{\bar{x} \mid \bar{x} \in A, x' \succ \bar{x}\}$

$visited(x') = \text{true}$

until $\nexists \tilde{x} \in A$ such that $visited(\tilde{x}) = \text{false}$

return A

Pareto Local Search Applying an LS in the form of iterative improvement algorithms to a MOP is relatively simple. This can be done by modifying the acceptance criterion, making use of dominance relation to compare solutions in the objective space, and keeping an archive of the best non-dominated solutions. With such modifications, iterative improvement algorithms under the CWAC search model can iteratively improve the current set of candidate

solutions in the archive by adding non-dominated neighboring solutions (Paquete et al., 2007). Such an algorithm can be seeded either by one single solution that may be generated randomly or by a set of candidate solutions generated by, for example, an exact algorithm. An example of iterative improvement algorithms for MOPs is the Pareto Local Search (PLS), proposed in Paquete et al. (2007). PLS algorithm is illustrated in Algorithm 6. Given an initial solution (x_0), PLS iteratively applies the following steps. First, it randomly chooses an unvisited solution x from the archive (A). Then, the neighborhood of x ($\text{Neighbors}(x)$) is thoroughly explored, and all neighbors that are not dominated by any solution in the archive are added to the archive. Solutions in the archive dominated by the newly added solutions are removed. Once the neighborhood of x is fully explored, x is marked as visited. The algorithm stops when all solutions in the archive have been visited, which indicates that a particular set of local minimum solutions was obtained (Paquete et al., 2007).

Algorithm 7 The general scheme of Pareto Iterated Local Search

procedure PARETO ITERATED LOCAL SEARCH:

 Generate initial solution x_0

$A = \text{PLS}(x_0)$

while termination criterion is not satisfied **do**

$x = \text{Choose}(A)$

$x' = \text{Perturb}(x)$

$A' = \text{PLS}(x')$

$A = \text{MergeFilter}(A, A')$

return A

Iterated Pareto Local Search A major problem of PLS algorithms is that they may get trapped in a set of local minima. Exactly in such a situation, an action should take place that allows the algorithm to leave the local minima set and to continue the search for possibly better solutions. One straightforward possibility is to *perturb* one or more solutions in the archive and restarts the local search from them. This is analogous to the Iterated Local Search described in (Lourenço et al., 2010) for single-objective optimization problems. We name this extension for MOPs, *Iterated Pareto Local Search (IPLS)*. Algorithm 7 shows the working principle of IPLS. First, it generates an initial solution (x_0), then it calls PLS with x_0 as an initial solution and keeps the non-dominated solutions in the archive A . In order to help the PLS to escape out from local optima, IPLS iterates by recalling the following

procedures. First, function Choose randomly selects a solution x from archive A . Then, function Perturb perturbs the solution x into x' . PLS is called again with x' to create a new archive A' . In the end, function MergeFilter add the new non-dominated solutions from A' into A and eliminates the dominated solutions from A . The algorithm continues until some termination criterion is met, such as the number of iterations or some kind of convergence is achieved.

3.4 Performance Assessment of Heuristic Methods

There are different aspects to consider when measuring the performance of different heuristic approaches for MOPs, such as the quality of the solutions returned by each heuristic as well as the computation time to achieve those solutions. In this section, we focus on quality aspects of the approximation set returned by the heuristics with a particular focus on standard comparison procedures that have been reported in the literature.

We also note that most heuristic methods for solving MOPs take randomized decisions during the search process. Therefore, if we apply them several times to the same problem instance, each time may result in a different approximation set. From a statistical point of view, the collection of all approximation set is the population of interest that we would like to characterize. As a result, to compare two methods in the same problem instance, we need to compare their corresponding approximation set samples. In principle, there exist two approaches in the literature that allow analyzing two or more heuristic approaches from a statistical point of view. The first approach transforms each approximation set into a real value using quality indicators. Then, the resulting indicator values are compared based on statistical testing procedures. The alternative approach is the *attainment function method*. In this method, the approximation set samples are summarized in terms of the *empirical attainment function*. In the following, we will review the two approaches in more details.

3.4.1 Indicator of Performance

By far, the most commonly used approach to measure the quality of an approximation set is to apply *unary quality indicators*. A unary quality indicator is a function that assigns each approximation set a real number. Therefore, we can compare two heuristic methods by comparing the indicator values of the approximation sets that resulted from each. In the following, we introduce two well-known indicators: the hypervolume indicator (I_{HV}) and the unary ϵ -indicator (I_ϵ). In the following, we assume that a *reference* set R of non-dominated

points is available. It can be the non-dominated set, the non-dominated points obtained from the union of all available approximation sets, or some upper or lower bound set.

The Hypervolume Indicator (I_{HV}). This indicator measures the size of the region dominated by the approximation set bounded below by some reference point that is strictly dominated by all elements of the approximation set (Zitzler and Thiele, 1998). Note that we can consider the relative hypervolume with respect to a reference set R . Given an approximation set A , the relative of hypervolume indicator is defined as

$$I_{RHV}(A) = I_{HV}(A)/I_{HV}(R) \quad (3.12)$$

where larger values correspond to a better quality with respect to this indicator (Zitzler et al., 2003). The left plot of Figure 3.9 illustrates the hypervolume indicator with two maximizing objectives. The reference set and approximation set is shown with white and gray points, respectively. For the given approximation set and a selected reference point, the hypervolume indicator for the approximation set corresponds to the size of the region shown in the white and the gray area plus the white corresponds to the hypervolume of the reference set. The main advantage of the hypervolume indicator is that it always increases as a non-dominated point is added to the approximation set, and its value is maximal if the approximation set coincides with the non-dominated point set. Similarly, the relative hypervolume indicator increases as a non-dominated point is added to the approximation set, and its value is maximum if the approximation set coincides with the non-dominated point set. Zitzler et al. (2003) has shown that the hypervolume indicator is one of the few that have these properties.

The Unary ϵ -indicator (I_ϵ) and ($I_{\epsilon+}$). Zitzler et al. (2003) introduced the ϵ -indicator family, a multiplicative and an additive version. The ϵ -indicator (multiplicative), $I_\epsilon(A, R)$, corresponds to the smallest factor ϵ that can be multiplied to each element in the approximation set, A , such that every point in the reference set, R , is weakly dominated by the resulting transformed approximation set that is

$$I_\epsilon(A, R) = \inf_{\epsilon \in \mathbb{R}} \{ \forall r \in R \exists a \in A : a \succeq_\epsilon r \} \quad (3.13)$$

where the ϵ -dominance relation, \succeq_ϵ , is defined as

$$u \succeq_\epsilon v \iff u_i \geq \epsilon \cdot v_i, \quad i = 1, \dots, q \quad (3.14)$$

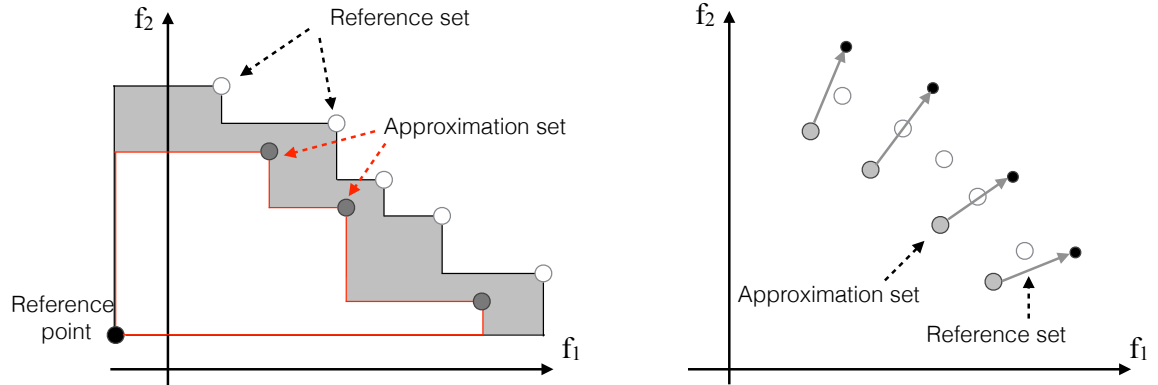


Figure 3.9: Illustration of the hypervolume indicator (left) and the ϵ -indicator (right)

for a maximization problem, and assuming that all non-dominated points are positive in all objectives. An equivalent unary additive epsilon indicator, $I_{\epsilon+}(A, R)$, is defined analogously with the additive ϵ -dominance $\succeq_{\epsilon+}$

$$u \succeq_{\epsilon+} v \iff u_i \geq \epsilon + v_i, i = 1, \dots, q \quad (3.15)$$

Figure 3.9 illustrates the ϵ -indicator. The approximation set and the reference set is indicated with gray and white points, respectively.

Note that, in some cases, the hypervolume and the ϵ -indicator may return different orderings of a given collection of approximation sets. This naturally arises since both indicators summarize the approximation sets from different points of view. There is an in-depth discussion about the relationship between these two indicators and others in (Zitzler et al., 2003) and (Knowles et al., 2006).

3.4.2 Attainment Function

Fonseca and Fleming (1996), and later developed in Shaw et al. (1999), da Fonseca et al. (2001) and Fonseca et al. (2005), introduced the concept of *attainment function* (AF) to characterize the statistical distribution of approximation sets in the objective space. This is particularly relevant for comparing and characterizing randomized heuristic methods. Given a collection of approximation sets obtained from a sequence of runs of a randomized heuristic, they can be partially characterized by the *empirical attainment function* (EAF), which can be seen as a generalization of the empirical probability distribution (Fonseca et al., 2005).

To explain the underlying idea, suppose a randomized heuristic is run on a specific problem, which produces an approximation set. For each objective point z in the objective

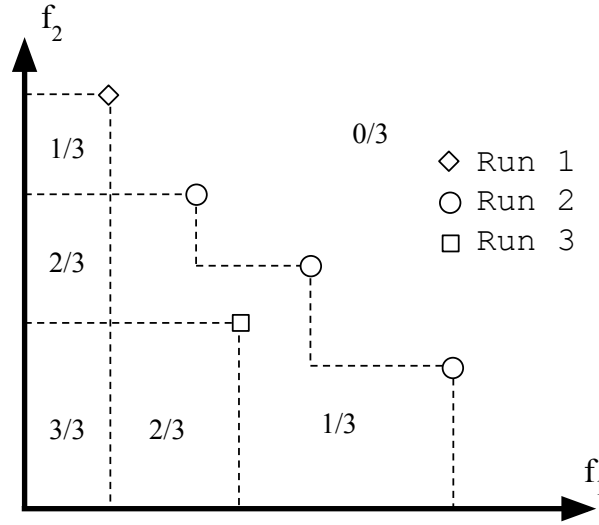


Figure 3.10: Illustration of the empirical attainment function

space, there is a probability P that the resulting approximation set contains a non-dominated point that weakly dominates z . We say that P is the probability that z is attained by the heuristic. The AF gives, for each objective point z in the objective space, the probability that z is attained in one run. The true attainment function is unknown, but we can estimate it, based on the approximation set collected from several runs by counting the number of approximation sets by which each objective point is attained and dividing it by the number of runs (sample size).

Figure 3.10 shows an example of three approximation sets collected from three runs and the corresponding empirical attainment function. The lower left region is attained in all runs and, therefore, is assigned to a relative frequency of 3/3; the upper right region is attained in none of the runs and the two remaining regions are assigned to 1/3 and 2/3 since they are attained in one and two of the three runs, respectively. A more formal definition is given by Fonseca et al. (2005) is as follows:

Definition 3.4.1. (*Attainment Function and Empirical Attainment Function*) Let $z \in \mathbb{R}^q$ and $q \geq 2$. Let $A = \{a_j \in \mathbb{R}^q, j = 1, \dots, m\}$ be a random set of m non-dominated points. The attainment function AF is defined as

$$AF(z) = P(a_1 \succeq z \vee a_2 \succeq z \vee \dots \vee a_m \succeq z) = P(A \succeq \{z\}) \quad (3.16)$$

The attainment function can be estimated from a sample of r random non-dominated point

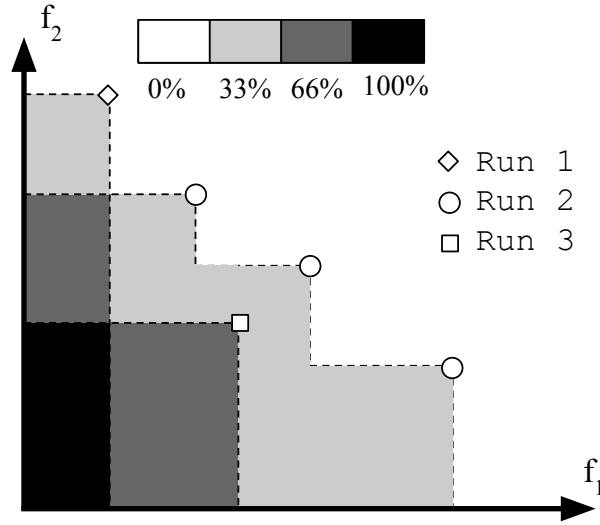


Figure 3.11: Hypothetical outcomes of the three runs for a MOP instance

sets through the empirical attainment function (EAF), defined as :

$$EAF(z) = \frac{1}{r} \sum_{i=1}^r I(A_i \succeq \{z\}) \quad (3.17)$$

where A_i is the i -th non-dominated point set and I is the indicator function.

EAFs is particularly useful for visualizing the outcomes of multiple runs of a heuristic. For instance, one might be interested in plotting all the points that have been attained in 50% of the runs or other quantiles. By running the heuristics for several times, we can produce several of these surfaces, which will lead us to have an estimation of the *quantiles* of the corresponding AF. These are boundaries in the objective space that delimit regions which are likely to be attained with the same probability. The $k\%$ attainment surface is an orthogonal polyline that separates the objective space attained by $k\%$ of the runs. For example, the median attainment surface delimits the region attained by 50% percent of the runs. Similarly, the worst attainment surface delimits the region attained by all runs, whereas the best attainment surface corresponds to the line between the region attained by at least one run and the points that were never attained by any run. Figure 3.11 shows an example of the EAF computed for three approximation sets. The lower left region is attained in all of the runs (black area) and therefore is assigned to 100%, the upper right region is attained in none of the runs (0%) and the remaining two regions are assigned 33% and 66% because

they are attained in one and two of the three runs, respectively. The polylines that separate the regions are the corresponding attainment surfaces.

`EAFtools` is an R script for computing bi-dimensional EAFs from a collection of approximation sets and plotting attainment surfaces as well as differences in terms of EAFs produced by pairs of heuristics (López-Ibañez et al., 2010). The package is available at <http://lopez-ibanez.eu/eaftools>.

3.5 Summary and Discussion

Many real-world problems, such as the sequence alignment problem involves the simultaneous optimization of several conflicting objectives. When dealing with MOPs, there is a set of alternative trade-offs, generally known as Pareto optimal set. These solutions are optimal on a broader sense that no other solutions in the feasible set are superior to them when all objectives are considered.

In this chapter, the principles of multiobjective optimization are outlined, and basic concepts are formally defined. This is followed by a discussion about traditional approaches to approximate the non-dominated point set. Afterwards, local search and evolutionary algorithms are presented as optimization methods which possess several characteristics that are desirable for this kind of problem. The performance assessment methods are briefly outlined in order to compare the results in this research area.

In the next chapter, one of the main contributions of this thesis is introduced, in particular, the biobjective formulation of the pairwise sequence alignment and algorithms to solve it.

Chapter 4

Multiobjective Pairwise Sequence Alignment

4.1 Introduction

Many problems that arise in Bioinformatics and Computational Biology can be formulated as multiobjective optimization problems (Handl et al., 2007). One of these problems is the sequence alignment. Typically, this problem has been tackled as a weighted sum optimization problem with weights for matches, mismatches and indels/gaps (see Section 2.3). Moreover, there is often a considerable disagreement on how to specify fixed values for these weights in the most commonly used sequence alignment software packages (Morrison, 2015). Multiobjective sequence alignment can overcome the problem of setting weights *a priori*, and it can also provide different alignments that may give further information to the practitioners. To the best of our knowledge, Roytberg et al. (1999) was the first to propose a multiobjective formulation of the pairwise sequence alignment and an algorithm to solve it. As opposed to many other multiobjective optimization problems (see discussion about the difficulty of solving these problems in Ehrgott (2000); Figueira et al. (2017)), this formulation leads to a polynomial number of non-dominated points and to solution approaches that take polynomial amount of time, which is not only an interesting result for itself but also an appealing result for the practitioner, since she does not need to investigate (*exponentially*) many alignments.

In the following, we describe the main contributions of this chapter, which resulted in the

publication of two journal articles (Abbasi et al., 2013a; Paquete et al., 2014), two extended abstracts (Paquete et al., 2012a,b) and two posters presented at international conferences (Abbasi et al., 2011, 2012).

- i) We extend the formulation of multiobjective pairwise sequence alignment given in (Roytberg et al., 1999) for the case of gaps and substitution matrices. In addition, we modified some steps of the dynamic programming algorithm given in (Roytberg et al., 1999), which improved its running time. Moreover, we introduce extensions of this approach to the other formulations.
- ii) We propose a novel pruning technique, based on the branch-and-bound principle, which considerably reduces the number of states to be visited by the dynamic programming algorithm. The experimental results report a speed-up of 80% and a saving of memory usage up to 90%.
- iii) We also propose a dynamic programming approach that uses the ϵ -constraint principle (see Section 3.3.1). This approach maximizes the substitution score while considering several constraint values on the number of indels/gaps.
- iv) We conduct a thorough computational analysis of these approaches on randomly generated data sets as well as on real data sets from the benchmark BAliBase version 3.0 (Thompson et al., 2005).

The remainder of this chapter is organized as follows: Section 4.2 provides the required notation and definitions for multiobjective pairwise sequence alignment; Section 4.3 explains the parametric sequence alignment problem which is closely related to our multiobjective formulation; Section 4.4 describes the proposed algorithms for the several variants of the multiobjective pairwise sequence alignment problem; in Section 4.5, we empirically evaluate the proposed models, and finally we present the main conclusions and discussion in Section 4.6.

4.2 Notation and Definitions

In this section, we explain the required notation and definitions for multiobjective pairwise sequence alignment. We start by defining the several problem variants.

Definition 4.2.1. Let φ be an alignment of two sequences $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$. Let $m(\varphi)$ and $d(\varphi)$ indicate the number of matches and indels respectively in the alignment φ and let $g(\varphi)$ and $s(\varphi)$ indicate the gap and substitution score of the alignment according to affine gap model and a substitution matrix M , respectively. The following (vector) score functions are defined:

$$VMD(\varphi) = (m(\varphi), -d(\varphi))$$

$$VMG(\varphi) = (m(\varphi), g(\varphi))$$

$$VSD(\varphi) = (s(\varphi), -d(\varphi))$$

$$VSG(\varphi) = (s(\varphi), g(\varphi))$$

Let Φ denote the set of all feasible alignments. The biobjective sequence alignment problem consists of finding the alignments that are “maximal” with respect to the above score functions. In the following, we define four biobjective sequence alignment problems, each of which related to a score function presented in Definition 4.2.1.

$$\arg \text{vmax} \{ \varphi : VMD(\varphi), \varphi \in \Phi \} \quad (\text{VMDP})$$

$$\arg \text{vmax} \{ \varphi : VMG(\varphi), \varphi \in \Phi \} \quad (\text{VMGP})$$

$$\arg \text{vmax} \{ \varphi : VSD(\varphi), \varphi \in \Phi \} \quad (\text{VSDP})$$

$$\arg \text{vmax} \{ \varphi : VSG(\varphi), \varphi \in \Phi \} \quad (\text{VSGP})$$

where $\arg \text{vmax}$ is understood in terms of Pareto optimality (see Section 3.2.1). The image of set Φ in the score function space is called *feasible score set*. Using the example of problem VMDP, the notion of Pareto optimality is described as follows: Given two feasible alignments φ and φ' , $VMD(\varphi) \succ VMD(\varphi')$ (φ dominates φ') if and only if it holds that $m(\varphi) \geq m(\varphi')$ and $d(\varphi) \leq d(\varphi')$, and $VMD(\varphi) \neq VMD(\varphi')$ (see Section 3.2.1, page 45). Accordingly, an alignment φ^* is *Pareto optimal* if there exists no other feasible alignment φ such that $VMD(\varphi) \succ VMD(\varphi^*)$. The set of all Pareto optimal alignments is called *Pareto optimal alignment set*. The image of a Pareto optimal alignment in the score function space is a *non-dominated score* and the set of all non-dominated scores is called *non-dominated score set*. The dominance relation and the above notation also apply to the other problem variants with the necessary changes.

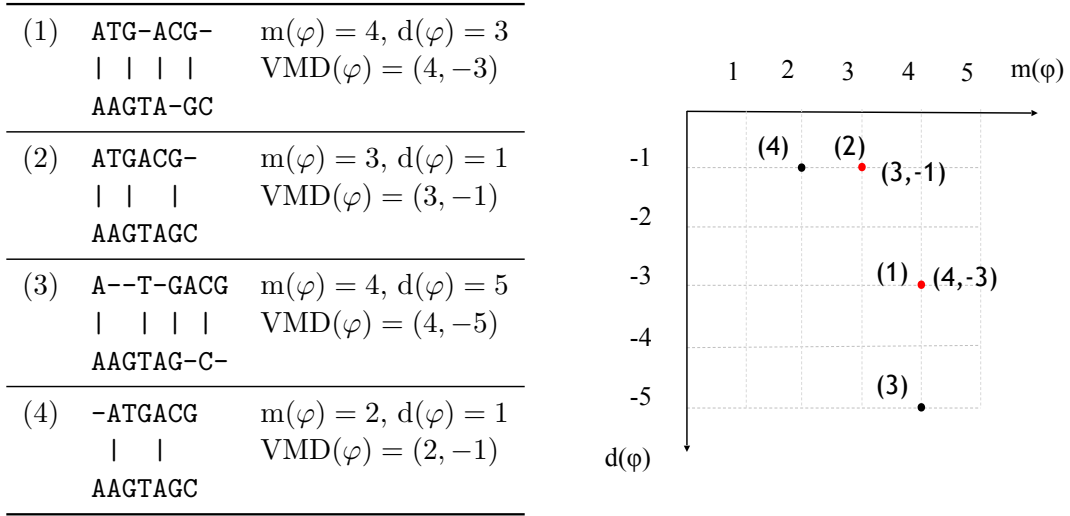


Figure 4.1: Various alignments of sequences AAGTAGC and ATGACG (left) and their images in the score space (right)

Figure 4.1 illustrates some possible alignments of two sequences AAGTAGC and ATGACG. Matches in the alignments are marked by vertical lines. Alignments (1) and (2) are Pareto optimal for the score function $VMD(\varphi)$. Note that alignment (1) dominates alignment (3) and alignment (2) dominates alignment(4). Besides, the scores of the alignments (1) and (2) are mutually on-dominated. In fact, for this example, the non-dominated score set is $\{(4, -3), (3, -1)\}$.

Computing the Pareto optimal alignment set can be an intractable task: Consider the sequences $A = \mathcal{G}^n$ and $B = \mathcal{T}(\mathcal{G}\mathcal{T})^{2n}$ and the substitution matrix $M[i, j] = 1$ and $M[i, j] = 0$, $i \neq j$; then, there exist $\binom{2n}{n}$ Pareto optimal alignments that match \mathcal{G}^n in both sequences since there exists no other alignment with larger substitution score and lesser number of indels ($3n + 1$ indels). The example above also applies to the score functions with gaps. However, the size of the non-dominated score set is bounded by $\min(\ell_{lcs}, n_1 + n_2 - 2\ell_{lcs})$, where ℓ_{lcs} is the size of longest common subsequence of A and B (see Roytberg et al., 1999). Note that few multiobjective combinatorial optimization problems are known to have this property (Figueira et al., 2017).

4.3 Parametric Sequence Alignment

In parametric sequence alignment, the problem of how to specify fixed values for the weights is avoided by computing the optimal alignment as a function of variable weights (Gusfield et al., 1994). This method partitions the weight space into convex regions such that in each region one alignment is optimal. Thus, parametric alignment allows understanding the effect of different weight combinations on the alignment. As presented in Eq. (2.3), for a simple sequence alignment, the weights α , β and μ can be varied to adjust the relative contributions of the matches, mismatches and indels.

In the simplest model of parametric sequence alignment (Gusfield et al., 1994), by considering that β has a fixed value, different values for (α, μ) correspond to different optimal alignments. In addition, a particular region in the (α, μ) -space may correspond to the same optimal alignment. (α, μ) -space can be decomposed into convex polygons such that any two points in the same polygon correspond to the same optimal alignment. Waterman (1994) and Gusfield et al. (1994) describe efficient algorithms for computing a polygonal decomposition of the weight space. (A recursive algorithm for an arbitrary number of objectives and for general multiobjective optimization problems is discussed in Przybylski et al. (2010).)

The parametric sequence alignment is, in fact, related to the notion of scalarization of a MOP (see Section 3.2.3). As a result of scalarization, these algorithms only return the set of supported solutions (see Section 3.3.1), which is a subset of a Pareto optimal set. Therefore, this approach may fail to find some relevant alignments.

4.4 Algorithms for Multiobjective Pairwise Alignment

For solving the multiobjective pairwise sequence alignment problem, two types of approaches are explained in the following sections: dynamic programming (DP) and ϵ -constraint. We assume that the aim is to find the non-dominated score set and not the set Pareto optimal alignment set, which may be exponentially large. The DP approach is based on the algorithm proposed in Roytberg et al. (1999), which extends the approach of Needleman and Wunsch (1970) (see Section 2.3.3, page 16). We improve the running time of this approach and introduce a pruning technique that is based on the comparison of lower and upper bounds, as performed in branch and bound procedures. The second method to solve the problem is based on the ϵ -constraint approach (Haimes et al., 1971) (see Section 3.3.1), which tackles the problem by solving a sequence of constrained sequence alignment problems.

4.4.1 Dynamic Programming for Multiobjective Pairwise Alignment

For two sequences of length n_1 and n_2 , Roytberg et al. (1999) introduced a multiobjective version of the DP approach in (Needleman and Wunsch, 1970) to solve the problem (VSDP) that takes $O(n_1 \cdot n_2 \cdot (n_1 + n_2) \log(n_1 + n_2))$ amount of time. In the following, we propose techniques to reduce its running time and extend it for solving Problem (VSGP) (see Section 4.2). The same techniques can be used for solving Problems (VMDP) and (VMGP) by a proper choice of values in the substitution matrix.

Improvements on the Dynamic Programming Algorithm for (VSDP)

The main idea to solve problem (VSDP) with DP is to compute the Pareto optimal alignments for all pairs of prefixes of the given sequences (Roytberg et al., 1999). The algorithm extends the DP approach for the single objective sequence alignment (see Section 2.3.3). The recursive formulation of the DP algorithm is similar to that of the single objective formulation. Similarly, the solution to each subproblem depends only on three smaller subproblems. However, at each step of the recursion, the set of non-dominated scores must be chosen.

In the following, we show that a multiobjective variant of the *optimal substructure* property holds for this problem,¹ that is, a Pareto optimal alignment for a subproblem can be obtained from a Pareto optimal alignment to one of the three immediately smaller subproblems; this is analogous to the proof of *optimal substructure* in Proposition 2.3.4.

Proposition 4.4.1. *Let $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$ be two sequences. Let $\varphi = (\varphi_1, \dots, \varphi_\ell)$ be a Pareto optimal alignment of A and B with score $(s(\varphi), -d(\varphi))$.*

- i) If $\varphi_\ell = (a_{n_1}, b_{n_2})$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is a Pareto optimal alignment of (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{n_2-1}) ;*
- ii) if $\varphi_\ell = (a_{n_1}, '-')$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is a Pareto optimal alignment of (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{n_2}) ;*
- iii) if $\varphi_\ell = ('-', b_{n_2})$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is a Pareto optimal alignment of (a_1, \dots, a_{n_1}) and (b_1, \dots, b_{n_2-1}) .*

Proof.

case i) Assume that $(\varphi_1, \dots, \varphi_{\ell-1})$ is not a Pareto optimal alignment for prefixes (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{n_2-1}) . Then, there exists another Pareto optimal alignment, φ' , for the

¹The proof of correctness of the DP algorithm was not shown in (Roytberg et al., 1999).

same prefixes with score $(s(\varphi'), -d(\varphi')) \succ (s((\varphi_1, \dots, \varphi_{\ell-1}), -d((\varphi_1, \dots, \varphi_{\ell-1})))$. By summing up the score vector $(M[a_i, b_j], 0)$ to $(s(\varphi'), -d(\varphi'))$, we obtain a new score vector that dominates $(s(\varphi), -d(\varphi))$, which is a contradiction;

case ii) Assume that $(\varphi_1, \dots, \varphi_{\ell-1})$ is not a Pareto optimal alignment for prefixes (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{j-1}) . Then, there exists another Pareto optimal alignment, φ'' , for the same prefixes with score $(s(\varphi''), -d(\varphi'')) \succ (s((\varphi_1, \dots, \varphi_{\ell-1}), -d((\varphi_1, \dots, \varphi_{\ell-1})))$. By summing up the score vector $(0, -1)$ to $(s(\varphi''), -d(\varphi''))$, we obtain a new score vector that dominates $(s(\varphi), -d(\varphi))$, which is a contradiction;

case iii) Symmetric to ii).

□

Based on this proposition, we arrive to the following recursive definition for the sub-problem of finding the non-dominated score set of (a_1, \dots, a_i) and (b_1, \dots, b_j) , $1 \leq i \leq n_1$, $1 \leq j \leq n_2$. Let $\Gamma(i, j)$ denote the non-dominated score set of two prefixes (a_1, \dots, a_i) and (b_1, \dots, b_j) , $1 \leq i < n_1$, $1 \leq j < n_2$.

$$\Gamma(i, j) = \begin{cases} \{(0, -i)\} & \text{if } i = 0 \\ \{(0, -j)\} & \text{if } j = 0 \\ \text{vmax} \begin{cases} \{\gamma + (M[a_i, b_j], 0) : \gamma \in \Gamma(i-1, j-1)\} \\ \{\gamma + (0, -1) : \gamma \in \Gamma(i-1, j)\} \\ \{\gamma + (0, -1) : \gamma \in \Gamma(i, j-1)\} \end{cases} & \text{if } i \neq 0 \text{ and } j \neq 0 \end{cases} \quad (4.1)$$

where $M[a_i, b_j]$ is the substitution score for (a_i, b_j) , and $\Gamma(0, j)$ and $\Gamma(i, 0)$ correspond to the base case of the recursion, that is, the score of aligning a prefix with an empty sequence. In the case that $i \neq 0$ and $j \neq 0$, operator vmax merges the three sets and extracts the states that are not dominated by any other state; note that $\Gamma(i, j)$ may contain states from the three sets. In the following, we show that $\Gamma(i, j)$ contains the non-dominated score set.

Proposition 4.4.2. *Recursion (4.1) computes the non-dominated score set.*

Proof. For the basis cases, we have that:

i) If $i = 0$ and $j = 0$, the non-dominated score of two empty sequences is $(0, 0)$;

- ii) If $i = 0$ and $0 < j \leq n_2$, which corresponds to aligning a non-empty sequence to an empty sequence, the only non-dominated score is $(0, -j)$;
- iii) Symmetric to ii).

If $i \neq 0$ and $j \neq 0$, then, from Proposition 4.4.1, the three sets contain all, but not only, the non-dominated score sets for $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$. Since vmax merges the three sets and keeps only those that are not dominated, set $\Gamma(i, j)$ contains all and only non-dominated scores. By induction, $\Gamma(n_1, n_2)$ contains all and only non-dominated scores for the original problem. \square

Similarly to the single-objective case, set $\Gamma(i, j)$ can be stored in a bi-dimensional matrix in order to discard recursive function calls. This would correspond to a *top-down* DP approach, which coincides with the approach in (Roytberg et al., 1999).

For a given alignment φ of two sequences $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$, we define a state $p = \text{VSD}(\varphi)$. A matrix \mathbf{P} is constructed for storing the non-dominated score set. Each entry $\mathbf{P}[i, j]$, for $(i, j) \in \{1, \dots, n_1\} \times \{1, \dots, n_2\}$, stores the list of states corresponding to the set of images of the Pareto optimal alignments of subsequences (a_1, \dots, a_i) and (b_1, \dots, b_j) . The recurrence for computing $\mathbf{P}[i, j]$ in a bottom-up fashion is as follows:

$$\mathbf{P}[i, j] = \text{vmax} \begin{cases} \{p + (M[a_i, b_j], 0) : p \in \mathbf{P}[i-1, j-1]\} \\ \{p + (0, -1) : p \in \mathbf{P}[i-1, j]\} \\ \{p + (0, -1) : p \in \mathbf{P}[i, j-1]\} \end{cases}$$

with the basis cases

$$\mathbf{P}[i, 0] = \{(0, -i)\} \quad 1 \leq i \leq n_1 \quad (4.2)$$

$$\mathbf{P}[0, j] = \{(0, -j)\} \quad 1 \leq j \leq n_2 \quad (4.3)$$

$$\mathbf{P}[0, 0] = \{(0, 0)\} \quad (4.4)$$

Assuming two sequences of length n , the approach above, as described in Roytberg et al. (1999) has $O(n^3 \log n)$ time-complexity, due to the number of cells in matrix \mathbf{P} , the number of non-dominated scores stored at each cell of \mathbf{P} and the time complexity of removing dominated scores at each iteration. In the following, we introduce some further improvements.

1. Faster removal of dominated scores Roytberg et al. (1999) suggests the use of a quasi-linear time algorithm for performing the operation of vmax by applying a sweep-line

approach after merging the three sets (see the general algorithm described in (Kung et al., 1975)). However, this can be performed in linear time under the assumption that each of the three sets is already sorted with respect to one of the objectives. Similar idea is also described in (Beier and Vöcking, 2011, pp. 380) for a different problem. In the following, each state $p = \text{VSD}(\varphi)$ is shown briefly with $p = (s, d)$ where $s = s(\varphi)$ and $d = -d(\varphi)$. Moreover, assume that the set of non-dominated scores in entry $\mathbf{P}[i-1, j-1]$, $\mathbf{P}[i, j-1]$ and $\mathbf{P}[i, j]$, $1 \leq i \leq n_1$, $1 \leq j \leq n_2$, is kept in a sorted list L_k , $k = 1, 2, 3$, respectively, in increasing order of the number of indels (and decreasing order of the substitution score). The algorithm iterates over the following steps to compute the non-dominated score set for entry $\mathbf{P}[i, j]$.

1. Set $s_{\max} = -\infty$ and indices $\ell_k = 1$, for $k = 1, 2, 3$;
2. For each list L_k , for $k = 1, 2, 3$, search linearly starting at index ℓ_k for the first state with score (s_k, d_k) such that $s_k > s_{\max}$; if no state is found for each of the three lists, return $\mathbf{P}[i, j]$; otherwise, let ℓ'_k be the index in L_k where the state with score (s_k, d_k) was found.
3. Select one of the three states with the least number of indels and, in case of equality, that with the maximum substitution score s ; append it to $\mathbf{P}[i, j]$.
4. Set $s_{\max} = s$ and $\ell_k = \ell'_k$, $k = 1, 2, 3$, and go to 2.

Note that s_{\max} is used for advancing the iterators in each list, passing through dominated states, until a state with a substitution score s larger than s_{\max} is found, which is a potential non-dominated state. Since the lists are ordered, a non-dominated state among those found in the lists at each step is probably a non-dominated state for the subproblem, since there exists no other in the three lists that can dominate it. The algorithm continues until all the states in the three lists are visited. Therefore, the overall time-complexity of this operation is $O(n_1 + n_2)$. Hence, the overall time and space-complexity of the proposed algorithm for Problem (VSDP) is $O(n_1 \cdot n_2 \cdot (n_1 + n_2))$, which improves upon the approach described in (Roytberg et al., 1999).

2. Pruning strategy in dynamic programming We describe a pruning technique for the DP algorithm for Problem (VSDP) that is able to reduce the number of states by comparing their upper bounds with a pre-computed lower bound set, following a branch-and-bound principle.

- *Lower bound set.* For the definition of lower bounds on the non-dominated score set, we use the notions of *lexicographic* and *scalarized* score functions as in Section 3.2.2 and 3.2.3, respectively.

We consider two lexicographic optimization problems that consist of finding an alignment that is lexicographic maximal (lexmax) according to a given order of priority on the optimization of the two score function components:

$$\arg \text{lexmax} \{ \varphi : (s(\varphi), -d(\varphi)), \varphi \in \Phi \} \quad (\text{LexSDP})$$

$$\arg \text{lexmax} \{ \varphi : (-d(\varphi), s(\varphi)), \varphi \in \Phi \} \quad (\text{LexDSP})$$

The order of the function components indicates the priority that is considered among the objectives. Let φ_s and φ_d be the lexicographic maximal alignments for Problems (LexSDP) and (LexDSP), respectively. Let $\text{MAX} = \text{VSD}(\varphi_s)$ and $\text{MIN} = \text{VSD}(\varphi_d)$. By the definition of optimality for Problem (VSDP), it holds that MAX and MIN belong to the non-dominated score set (Ehrgott, 2005). Moreover, they indicate that there cannot exist a Pareto optimal alignment with more (less) substitution score and indels than given by the components of MAX (MIN). Hence, the two score vectors give a bound on the possible ranges of the non-dominated score set. In fact, if $\text{MAX} = \text{MIN}$, then the non-dominated score set contains only a single element and no further computation is required.

Another lower bound is given by the solution to a *scalarized* version of the biobjective alignment problem. (see Section 3.2.3.) We consider the following weighted sum scalarization

$$\text{WSD}(\varphi) = w_s \cdot s(\varphi) + w_d \cdot d(\varphi)$$

where w_s and w_d are real weighting coefficients. The goal is to find the alignment that maximizes the scalarized score function as follows:

$$\arg \max \{ \varphi : \text{WSD}(\varphi), \varphi \in \Phi \} \quad (\text{WSDP})$$

Note that other scalarized functions are also possible. In the particular case of the weighted sum function, the alignment that is optimal to Problem (WSDP) is also Pareto optimal to Problem (VSDP), although the opposite does not hold in general

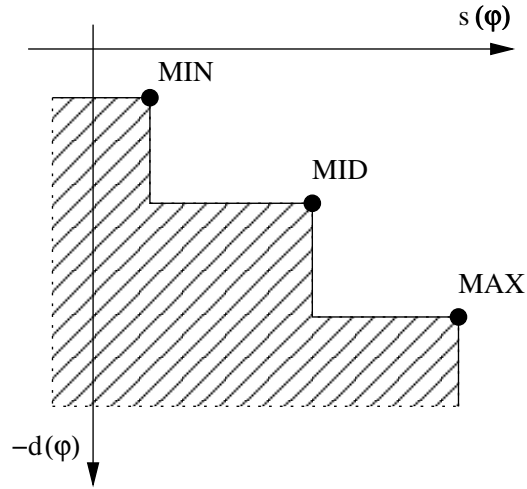


Figure 4.2: Illustration of the lower bound set. Lower bound set contains the three pre-computed non-dominated score vectors MIN, MID and MAX. Any alignment whose score vector is inside of the shaded area cannot be Pareto optimal, since it would be dominated by MIN, MID or MAX

(see Section 3.3.1, page 49). Let φ_w denote the optimal alignment for Problem (WSDP) for a given w_s and w_d and let $\text{MID} = \text{VSD}(\varphi_w)$. It is also important to highlight that the scalarized problem can be solved several times for different weights to get a tighter lower bound set; for the sake of the explanation, we use only one. The score vectors MAX, MID and MIN allow to define a lower bound set on the non-dominated score set of Problem (VSDP). Let \mathcal{R} denote the region

$$\mathcal{R} = \{r \in \mathbb{R}_0^+ \times \mathbb{R}_0^- : b \succ r, b \in \{\text{MAX}, \text{MID}, \text{MIN}\}\}.$$

Figure 4.2 illustrates the location of MAX, MID and MIN and definition of \mathcal{R} (shaded area). Note that there may exist further Pareto optimal alignments whose score vectors are located in the complement of \mathcal{R} . However, any alignment whose score vector is in the interior of \mathcal{R} cannot be Pareto optimal, since it would be dominated by an alignment with a score vector equals to MIN, MID or MAX.

The computation of the three score vectors can be performed with Needleman-Wunsch algorithm (see Section 2.3.3) by keeping the components separately in the DP matrix also, by choosing the state at each entry that maximizes the scalarized score function WSD, for the case of MID, or according to the lexicographic ordering for the case

of MAX and MIN, respectively. Therefore, the three score vectors can be found in $O(n_1 \cdot n_2)$ -time. In the following, we only introduce the recurrence formula that is required for computing the lexicographic maximal alignment for Problem (LexSDP), which gives the score vector MAX. We consider a DP matrix \mathbf{L} , where for each entry $\mathbf{L}[i, j]$, $(i, j) \in \{1, \dots, n_1\} \times \{1, \dots, n_2\}$, stores the state corresponding to a lexicographic maximal alignment of subsequences (a_1, \dots, a_i) and (b_1, \dots, b_j) . The elements of matrix \mathbf{L} are calculated recursively as follows:

$$\mathbf{L}[i, j] = \text{lexmax} \begin{cases} \mathbf{L}[i-1, j-1] + (M[a_i, b_j], 0) \\ \mathbf{L}[i, j-1] + (0, -1) \\ \mathbf{L}[i-1, j] + (0, -1) \end{cases}$$

with basis cases $\mathbf{L}[0, 0] = (0, 0)$, $\mathbf{L}[i, 0] = (0, -i)$ and $\mathbf{L}[0, j] = (0, -j)$, for $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$. The operator lexmax keeps only the lexicographic maximum of the three states in the recursive step.

- *Upper bound.* For a given state $t = (s, -d)$ at entry $\mathbf{P}[i, j]$, $(i, j) \in \{1, \dots, n_1\} \times \{1, \dots, n_2\}$, if its upper bound $\text{ub}(s) = (s + \theta, -d - \eta)$ is located in the interior of \mathcal{R} , then t will not lead to any state that corresponds to a non-dominated score vector. Therefore, state t can be discarded from entry $\mathbf{P}[i, j]$. We consider an upper bound of a state t in $\mathbf{P}[i, j]$ that is given by the maximum substitution score θ and minimum number of indels η that can be achieved from entry $\mathbf{P}[i, j]$ to entry $\mathbf{P}[n_1, n_2]$. The value of θ can be computed by the size of the *longest common subsequence* (with a substitution matrix) of $(a_{i+1}, \dots, a_{n_1})$ and $(b_{j+1}, \dots, b_{n_2})$. This can be easily obtained for every entry in matrix \mathbf{P} in a pre-processing step with the classical DP algorithm for computing the longest common subsequence in the reversed sequences. The minimum number of indels η is computed by the absolute difference between the sizes of two subsequences (a_i, \dots, a_{n_1}) and (b_j, \dots, b_{n_2}) , i.e. $\eta = |(n_2 - j) - (n_1 - i)|$. Therefore, $\text{ub}(s) = (s + \theta, -d - \eta)$ is a valid upper bound of state t . Note that this bound may not correspond to a feasible alignment. Matrix \mathbf{L} as well as the longest common subsequence for the reversed sequences can be computed in a pre-processing phase in $O(n_1 \cdot n_2)$ -time. Hence, the upper bound at each matrix entry can be computed in a constant amount of time during the main phase of the algorithm.

Table 4.1: Upper bounds for the example in Section 4.4.1

		A	C	T	A	G	G	G	
	0	1	2	3	4	5	6	7	
	0	(4,1)	(4,2)	(4,3)	(4,4)	(3,5)	(2,6)	(1,7)	(0,8)
A	1	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(2,5)	(1,6)	(0,7)
G	2	(3,1)	(3,0)	(3,1)	(3,2)	(3,3)	(2,4)	(1,5)	(0,6)
G	3	(3,2)	(3,1)	(2,0)	(2,1)	(2,2)	(2,3)	(1,4)	(0,5)
G	4	(3,3)	(3,2)	(2,1)	(1,0)	(1,1)	(1,2)	(1,3)	(0,4)
C	5	(3,4)	(3,3)	(2,2)	(1,1)	(1,0)	(1,1)	(1,2)	(0,3)
C	6	(2,5)	(2,4)	(2,3)	(1,2)	(1,1)	(1,0)	(1,1)	(0,2)
T	7	(1,6)	(1,5)	(1,4)	(1,3)	(1,2)	(1,1)	(1,0)	(0,1)
G	8	(0,7)	(0,6)	(0,5)	(0,4)	(0,3)	(0,2)	(0,1)	(0,0)

In the following, we describe an example that illustrates the pruning technique for Problem (VSDP). Let $A = \text{AGGGCCTG}$ and $B = \text{ACTAGGG}$. Table 4.2 (left) shows the contents of matrix \mathbf{P} without using the pruning technique, where the non-dominated score set is given at entry $\mathbf{P}[8, 7]$. The lower bounds are $\text{MIN} = (-3, -1)$ and $\text{MAX} = (4, -7)$; for $w_s = 1.5$ and $w_d = -0.5$, we have that $\text{MID} = (3, -5)$. Table 4.1 gives the upper bound for each entry (i, j) , i.e. the maximum substitution score and the minimum number of indels for the suffixes (a_i, \dots, a_{n_1}) and (b_j, \dots, b_{n_2}) . Table 4.2 (right) shows matrix \mathbf{P} with the pruning technique; each dash indicates the location of a state that was pruned. For example, the upper bound of state $t = (0, -6)$ at entry $\mathbf{P}[6, 0]$ is $\text{ub}(t) = (0 + 2, -6 - 5) = (2, -11)$, which is dominated by MAX and MID . Therefore, this state can be pruned. In contrast, for the state $t = (1, -2)$ at entry $\mathbf{P}[5, 3]$ we have that $\text{ub}(s) = (1 + 1, -2 - 1) = (2, -3)$, which is not dominated by any lower bound. Hence, in this case, state t cannot be pruned.

A Dynamic Programming Algorithm for (VSGP)

In this section, we introduce a (bottom-up) DP algorithm for Problem (VSGP). This approach extends the DP algorithm for global alignment with gap penalties for the affine gap model (see Section 2.3.4, page 21). For a given alignment $\varphi = (A', B')$, we define a state $q = \text{VSG}(\varphi)$ (see Section 4.2). For computing the set of non-dominated scores, we keep four dynamic programming matrices: \mathbf{Q} , \mathbf{R} , \mathbf{S} , and \mathbf{T} . For a given $(i, j) \in \{1, \dots, n_1\} \times \{1, \dots, n_2\}$, entry $\mathbf{R}[i, j]$, $\mathbf{S}[i, j]$ and $\mathbf{T}[i, j]$ stores the set of non-dominated scores corresponding to Pareto optimal alignments of prefixes (a_1, \dots, a_i) and (b_1, \dots, b_j) that end with (a_i, b_j) , $(\text{---}, b_j)$

Table 4.2: Matrix **P** without (left) and with (right) pruning for the example in Section 4.4.1

	A	C	T	A	G	G	G		A	T	A	G	G	G		
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	(0,0)	(0,-1)	(0,-2)	(0,-3)	(0,-4)	(0,-5)	(0,-6)	(0,-7)	(0,0)	(0,-1)	(0,-2)	(0,-3)	-	-	-	-
A 1	(0,-1)	(1,0)	(1,-1)	(1,-2)	(1,-3)	(1,-4)	(1,-5)	(1,-6)	(0,-1)	(1,0)	(1,-1)	(1,-2)	(1,-3)	-	-	-
G 2	(0,-2)	(1,-1)	(0,0)	(0,-1)	(0,-2)	(2,-3)	(2,-4)	(2,-5)	(0,-2)	(1,-1)	(0,0)	(0,-1)	(0,-2)	(2,-3)	-	-
		(1,-2)	(1,-3)	(1,-4)					(1,-2)	(1,-3)	(1,-4)					
G 3	(0,-3)	(1,-2)	(0,-1)	(-1,0)	(-1,-1)	(1,-2)	(3,-3)	(3,-4)	(0,-3)	(1,-2)	(0,-1)	(-1,0)	(-1,-1)	(1,-2)	(3,-3)	-
		(1,-3)	(0,-2)	(0,-3)	(2,-4)				(1,-3)	(0,-2)	(1,-3)	(0,-2)	-	(2,-4)		
		(1,-4)	(1,-5)	(1,-4)					(1,-4)	(1,-4)	-					
G 4	(0,-4)	(1,-3)	(0,-2)	(-1,-1)	(-2,0)	(0,-1)	(2,-2)	(4,-3)	-	(1,-3)	(0,-2)	(-1,-1)	(-2,0)	(0,-1)	(2,-2)	(4,-3)
		(1,-4)	(0,-3)	(-1,-2)	(1,-3)	(3,-4)			(1,-4)	(0,-3)	(1,-4)	(0,-3)	(-1,-2)	-	(3,-4)	
		(1,-5)	(0,-4)	(2,-5)					(1,-5)	(0,-4)	(2,-5)	-				
		(1,-6)	(1,-6)						(1,-6)							
C 5	(0,-5)	(1,-4)	(2,-3)	(-1,-2)	(-2,-1)	(-3,0)	(-1,-1)	(1,-2)	-	(1,-4)	(2,-3)	(-1,-2)	(-2,-1)	(-3,0)	(-1,-1)	-
		(2,-4)	(-1,-3)	(2,-4)	(2,-3)	(4,-4)			(2,-4)	(-1,-3)	(0,-2)	(2,-4)	(-1,-3)	(0,-2)	(2,-3)	(4,-4)
		(2,-5)	(1,-4)	(3,-5)	(3,-5)				(2,-5)	(1,-4)	(2,-5)	(2,-5)	(2,-5)	-	(3,-5)	
		(2,-6)	(2,-6)						(2,-6)							
C 6	(0,-6)	(1,-5)	(2,-4)	(1,-3)	(-2,-2)	(-3,-1)	(-4,0)	(-2,-1)	-	(1,-5)	(2,-4)	(-2,-2)	(-3,-1)	(-4,0)	(-2,-1)	-
		(2,-5)	(1,-4)	(0,-3)	(-1,-2)	(1,-3)			(2,-5)	(1,-4)	(0,-3)	-	(0,-3)	(-1,-2)	-	-
		(2,-6)	(1,-5)	(2,-4)	(4,-5)				(2,-6)	(1,-5)	(2,-4)	(2,-4)	(2,-4)	(4,-5)		
		(2,-7)	(3,-6)						(2,-7)	(3,-6)						
T 7	(0,-7)	(1,-6)	(2,-5)	(3,-4)	(0,-3)	(-3,-2)	(-4,-1)	(-5,0)	-	(1,-6)	(2,-5)	(3,-4)	(-3,-2)	(-4,-1)	-	-
		(3,-5)	(0,-4)	(-1,-3)	(2,-2)				(3,-5)	(0,-4)	(-1,-3)	(3,-5)	(-1,-3)	(2,-2)	(2,-2)	
		(3,-6)	(3,-6)	(2,-5)	(1,-4)				(3,-6)	(3,-6)	(2,-5)	(3,-6)	(2,-5)	(1,-4)	-	-
		(3,-7)	(4,-6)						(3,-7)	(4,-6)		(3,-7)	(4,-6)	(4,-6)		
G 8	(0,-8)	(1,-7)	(2,-6)	(3,-5)	(2,-4)	(1,-3)	(-2,-2)	(-3,-1)	-	(1,-7)	(2,-6)	(3,-5)	(-2,-2)	(-3,-1)	(-3,-1)	-
		(3,-6)	(4,-5)	(1,-4)	(0,-3)				(3,-6)	(4,-5)	(1,-4)	(1,-4)	(4,-5)	(0,-3)	(0,-3)	
		(4,-6)	(3,-5)	(4,-6)	(3,-5)				(4,-6)	(3,-5)	(4,-6)	(4,-6)	(3,-5)	(3,-5)	(3,-5)	
		(4,-7)	(4,-7)						(4,-7)					(4,-7)	(4,-7)	

and $(a_i, ' -')$, respectively, where $' -'$ is an indel character. The entry $\mathbf{Q}[i, j]$ stores the states corresponding to non-dominated scores of prefixes (a_1, \dots, a_i) and (b_1, \dots, b_j) . Let the substitution matrix be denoted by M , the gap opening penalty be ν and the gap extension be μ . Then, the recursion is as follows:

$$\begin{aligned} \mathbf{Q}[i, j] &= \text{vmax} \begin{cases} \mathbf{R}[i, j] \\ \mathbf{S}[i, j] \\ \mathbf{T}[i, j] \end{cases} \\ \mathbf{R}[i, j] &= \{q + (M[a_i, b_j], 0) : q \in \mathbf{Q}[i-1, j-1]\} \\ \mathbf{S}[i, j] &= \text{vmax} \begin{cases} \{q + (0, \mu) : q \in \mathbf{S}[i, j-1]\} \\ \{q + (0, \nu) : q \in \mathbf{Q}[i, j-1]\} \end{cases} \\ \mathbf{T}[i, j] &= \text{vmax} \begin{cases} \{q + (0, \mu) : q \in \mathbf{T}[i-1, j]\} \\ \{q + (0, \nu) : q \in \mathbf{Q}[i-1, j]\} \end{cases} \end{aligned} \tag{4.5}$$

The bases cases of the matrices are: $\mathbf{Q}[0, 0] = \{(0, 0)\}$, $\mathbf{Q}[i, 0] = \mathbf{S}[i, 0] = \mathbf{Q}[0, j] = \mathbf{T}[0, j] = \{(0, \mu)\}$, for $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$. Operation vmax takes also linear amount of time by using the same technique introduced for Problem (VSDP). Since the number of gaps is bounded from above by the number of indels in an alignment, the time and space-complexity is also $O(n_1 \cdot n_2 \cdot (n_1 + n_2))$.

Note that the properties of the optimal substructure for Problem (VSGP) hold similarly to Problem (VSDP). Moreover, this algorithm calculates the non-dominated score set by considering the affine gap model. By fixing the value of ν to 1 and μ to 0, this algorithm can solve the problem with constant gap model. When $\nu = \mu$, this algorithm is appropriate for the linear gap model. In fact, when $\nu = \mu = 1$, the algorithm becomes the DP algorithm for Problem (VSDP) (see Section 4.4.1, page 76).

For this problem, the computation of lower and upper bounds follow the same reasoning as for Problem (VSDP). In the following, we will only give a brief explanation and highlight the main differences.

- *Lower bound set.* The lexicographic and scalarized alignment problems described in the previous section (page 79) can also be formalized in terms of gaps. In this case, MAX and MIN correspond to the scores of Pareto optimal alignments that maximize

the substitution score and minimize the gap score, respectively. Also, MID corresponds to the score vector of the Pareto optimal alignment that maximizes a scalarized score function by taking into account the gap score (we will use w_g instead of w_d). Score vectors MAX and MID can be computed in $O(n_1 \cdot n_2)$ -time by using the algorithm described in (Gusfield, 1999) with the necessary changes. However, the score vector MIN can be computed faster. Note that the gap score can only be minimum when there exists zero or one gap, the latter case arising when $n_1 \neq n_2$. Assume, without loss of generality, that $n_1 < n_2$. Then, the computation of the maximum substitution score that is possible for one gap can be performed by comparing the substitution score for each of the $n_1 + 1$ possible locations of the gap. This can be performed in $O(n_2)$ -time in an incremental manner.

- *Upper bound.* The computation of the maximum substitution score that can be achieved at entry $\mathbf{Q}[n_1, n_2]$ by a state t at entry $\mathbf{Q}[i, j]$ follows the same procedure as explained in the previous section (page 82). For the computation of the minimum gap score, we consider a partition of matrix \mathbf{Q} into three sections; without loss of generality, we assume that $n_1 < n_2$. Let $\mathbf{Q}^D = \mathbf{Q}[i - n_2 + n_1, i]$, for $n_2 - n_1 \leq i \leq n_2$, which corresponds to the diagonal in \mathbf{Q} starting at $\mathbf{Q}[0, n_2 - n_1]$ and ending at $\mathbf{Q}[n_1, n_2]$. Let \mathbf{Q}^A and \mathbf{Q}^B denote the entries in matrix \mathbf{Q} that are located above and below \mathbf{Q}^D , respectively. From this partitioning of \mathbf{Q} , we can derive the following results for v , the minimum gap score, that is achieved at $\mathbf{Q}[n_1, n_2]$ by a state t (here, we relate state t with a partial alignment $\varphi = (A', B')$):

- i) If state $t \in \mathbf{Q}^D$, then $v = 0$;
- ii) If state $t \in \mathbf{Q}^A$ (\mathbf{Q}^B) and alignment φ ends with a gap in A' (B'), then $v = 0$;
- iii) If state $t \in \mathbf{Q}^A$ (\mathbf{Q}^B) and alignment φ ends with two characters or a gap character in B' (A'), then there is a gap with the size of $|(n_2 - j) - (n_1 - i)|$, where the minimum gap score in affine gap model can be computed simply as follows :

$$\eta = \nu + \mu \cdot |(n_2 - j) - (n_1 - i) - 1|$$

Note that conditions ii) and iii) can be determined by keeping an additional variable that stores whether state t was obtained from matrix \mathbf{R} , \mathbf{S} , or \mathbf{T} in the recursion. Therefore, the upper bound for the case of gaps can also be computed in a constant amount of time. Moreover, by fixing the value of ν to 1 and μ to 0, the above formula

calculates the upper bound for the constant gap model and, when $\nu = \mu$, it becomes the solution for the linear gap model.

4.4.2 An ϵ -constraint Algorithm for Multiobjective Pairwise Alignment

In this section, we introduce an ϵ -constraint technique (see Section 3.3.1) for solving problems (VSDP) and (VSGP). By changing the substitution matrix, the same techniques can be used for solving the Problems (VMDP) and (VMGP) as well. Our approach is based on the algorithm proposed in (Sankoff, 1972) for computing the sequence alignment that maximizes the number of matches with an inequality constraint on the number of gaps. In this work, we apply the same technique for solving Problems (VSDP) and (VSGP) with an equality constraint on the number of indels/gaps.

ϵ -constraint for Problem (VSDP)

The principle of ϵ -constraint method is based on optimizing (maximizing) one of the objectives while the others are used as constraints (see Section 3.3.1, page 50). To tackle Problem (VSDP) with an ϵ -constraint method, we consider the maximization of the substitution score with a constraint on the number of indels. Note that the maximum number of indels in an alignment is bounded by a non-negative integer (d_{\max}). Therefore, it is possible to solve Problem (VSDP) by maximizing the substitution score constrained to a given number of indels that range from 0 to d_{\max} , with an increment of one. From the set of optimal scores for all the $d_{\max} + 1$ constraint problems, the non-dominated score set is extracted by removing the dominated scores from that set.

Let $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$ be two sequences. Let ℓ_{cs} be the longest common subsequence between the two sequences; the maximal feasible number of indels is given by $d_{\max} = n_1 + n_2 - 2\ell_{cs}$. In the following, we show that optimal substructure also holds for the problem of maximizing substitution score given a fixed number of indels. We recall that, according to Proposition 2.3.2, any alignment between two sequences can possibly finish in three ways : i) (a_{n_1}, b_{n_2}) , ii) $(\text{' - '}, b_{n_2})$ or iii) $(a_{n_1}, \text{' - '})$.

Proposition 4.4.3. *Let $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$ be two sequences and let M be a substitution score matrix. Let $\varphi = (\varphi_1, \dots, \varphi_\ell)$ be an optimal alignment of A and B with maximum substitution score $s(\varphi)$ and k indels.*

- i) *if $\varphi_\ell = (a_{n_1}, b_{n_2})$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is an optimal alignment of (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{n_2-1}) with maximum substitution score $s(\varphi) - M[a_{n_1}, b_{n_2}]$ and k indels;*

- ii) if $\varphi_\ell = (a_{n_1}, ' - ')$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is an optimal alignment of (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{n_2}) with maximum substitution score $s(\varphi)$ and $k - 1$ indels;
- iii) if $\varphi_\ell = (' - ', b_{n_2})$, then $(\varphi_1, \dots, \varphi_{\ell-1})$ is an optimal alignment of (a_1, \dots, a_{n_1}) and (b_1, \dots, b_{n_2-1}) with maximum substitution score $s(\varphi)$ and $k - 1$ indels.

Proof.

- i) Assume that $(\varphi_1, \dots, \varphi_{\ell-1})$ is not an optimal alignment with maximum substitution score. Therefore, there exists another optimal alignment with larger substitution score for the same subsequences. By summing up the score of $M[a_{n_1}, b_{n_2}]$, an alignment with larger substitution score than $s(\varphi)$ is obtained, which is a contradiction. Now, assume that $(\varphi_1, \dots, \varphi_{\ell-1})$ is an optimal alignment that does not contain k indels. Consequently, by adding the pair (a_{n_1}, b_{n_2}) , it is not possible to have k indels in φ , which is, again, a contradiction;
- ii) Consider that $\varphi_\ell = (a_{n_1}, ' - ')$ and assume that there exists another optimal alignment with larger score than $s(\varphi)$ for (a_1, \dots, a_{n_1-1}) and (b_1, \dots, b_{n_2}) . This is a contradiction, since by adding the pair $(a_{n_1}, ' - ')$, there exists another alignment for A and B with larger score than $s(\varphi)$. Now assume that there exists another optimal alignment with more (less), than $k - 1$ indels for the same subsequences. By adding the pair $(a_{n_1}, ' - ')$ to the alignment, we have an alignment with more (less) indels than k , which is a contradiction.
- iii) Symmetric to ii).

□

By induction, it is possible to arrive in a general recursive definition to generate the optimal alignment. For each value of k , two matrices are required: Matrix \mathbf{P} , which keeps the maximum substituting score corresponding to the optimal alignment of subsequences (a_1, \dots, a_i) and (b_1, \dots, b_j) constrained to k indels that ends with the pair (a_i, b_j) , and matrix \mathbf{Q} , which keeps the maximum substituting score for the alignments of subsequences that end with $(a_i, ' - ')$ or $(' - ', b_j)$ and constrained with k indels. The recurrence for matrices $\mathbf{P}[i, j, k]$ and $\mathbf{Q}[i, j, k]$ for $(i, j, k) \in \{1, \dots, n_1\} \times \{1, \dots, n_2\} \times \{1, \dots, d_{max}\}$ are as follows:

$$\mathbf{P}[i, j, k] = \max \begin{cases} \mathbf{Q}[i - 1, j - 1, k] + M[a_i, b_j] \\ \mathbf{P}[i - 1, j - 1, k] + M[a_i, b_j] \end{cases} \quad (4.6)$$

$$\mathbf{Q}[i, j, k] = \max \begin{cases} \mathbf{Q}[i - 1, j, k - 1] \\ \mathbf{Q}[i, j - 1, k - 1] \\ \mathbf{P}[i - 1, j, k - 1] \\ \mathbf{P}[i, j - 1, k - 1] \end{cases} \quad (4.7)$$

The basis cases for $0 \leq i \leq n_1$, $0 \leq j \leq n_2$ and $0 \leq k \leq d_{\max}$ are:

$$\mathbf{Q}[i, j, 0] = -\infty \quad (4.8)$$

$$\mathbf{P}[i, j, 0] = \begin{cases} 0 & \text{if } i = j = 0 \\ P[i - 1, j - 1, 0] + M[a_i, b_j] & \text{if } i = j \neq 0 \\ -\infty & \text{if } i \neq j \end{cases} \quad (4.9)$$

$$\mathbf{P}[i, 0, k] = \mathbf{P}[0, j, k] = -\infty \quad k > 0 \quad (4.10)$$

$$\mathbf{Q}[i, 0, k] = \begin{cases} 0 & \text{if } i = k \\ -\infty & \text{if } i \neq k \end{cases} \quad (4.11)$$

$$\mathbf{Q}[0, j, k] = \begin{cases} 0 & \text{if } j = k \\ -\infty & \text{if } j \neq k \end{cases} \quad (4.12)$$

where the value $-\infty$ corresponds to an infeasible alignment and value 0 is used when there is no residue-residue pair² in the alignment. Eq. (4.8) gives the initial values for matrix \mathbf{Q} with $k = 0$. In this case, the alignments have to end with an indel-residue pair, but there is no possibility of having indels ($k = 0$); since they are infeasible, the matrix \mathbf{Q} takes the value $-\infty$. Eq. (4.9) gives the initial values for matrix \mathbf{P} with $k = 0$. In this case, the alignments have to end with a residue-residue pair while no indel is allowed in the alignment. Therefore, the only possible way to obtain a feasible alignment under these conditions is to “write down” one sequence below the other. Therefore, matrix \mathbf{P} can only take feasible

²A residue-residue pair is any pair in the alignment that does not contain indel.

values on the diagonal, which corresponds to align prefixes with equal sizes. It is clear that with $i = 0$, $j = 0$ and $k = 0$, the maximum substitution score is 0, and with $i = j > 0$ this score is the sum of the individual substitution scores of the previous residue-residue pairs in the alignment.

Eq. (4.10) gives the initial values for the first rows and columns of matrix \mathbf{P} , i.e., when $i = 0$ or $j = 0$ for all $0 < k \leq d_{\max}$. This corresponds to alignments between a sequence and an empty sequence while ending with a residue-residue pair. Since these alignments are infeasible, matrix \mathbf{P} takes the value $-\infty$. Eq. (4.11) gives the initial values for the first rows of matrix \mathbf{Q} . Using similar reasoning, in matrix \mathbf{Q} , the first rows when $i \neq k$ take $-\infty$ values. But when $i = k$, it is possible to align a sequence with an empty sequence by adding indels equal to the size of sequence. Finally, Eq. (4.12) works similarly to Eq. (4.11) for the first columns of matrix \mathbf{Q} .

For each iteration $k = 0, 1, \dots, d_{\max}$, the maximum between $\mathbf{P}[n_1, n_2, k]$ and $\mathbf{Q}[n_1, n_2, k]$ gives either the maximum score that can be achieved with k indels or $-\infty$ if no feasible alignment exists. To compute the non-dominated score set, these values need to be kept for each k . Due to the equality constraint, the algorithm finds a superset of the non-dominated score set. Therefore, once $k = d_{\max}$, a filtering step is needed to remove the dominated states. Note that, the corresponding Pareto optimal alignments can be obtained by tracing back the matrices just as in Needleman-Wunsch algorithm (see Section 2.3.3). In the following, we provide an illustrative example.

Table 4.3 shows the matrices \mathbf{P} and \mathbf{Q} with ($k = 0$) and ($k = 1$) for the example in Section 4.4.1, page 82. $\mathbf{P}[8, 7, 0] = \mathbf{Q}[8, 7, 0] = -\infty$ shows that there is no feasible alignment for the two sequences with no indel ($k = 0$). But with $k = 1$, the maximum between $\mathbf{P}[8, 7, 1] = -3$ and $\mathbf{Q}[8, 7, 1] = -5$ gives the maximum substitution score for an alignment with one indel. Therefore, $(-3, -1)$ is the first non-dominated score of the superset of non-dominated score set. The numbers shown in boxes inside matrices show the traceback path of a Pareto optimal alignment with the score $(-3, -1)$. The direction of movements in matrix \mathbf{P} is diagonally and in matrix \mathbf{Q} is either horizontally or vertically. Note that when the traceback is from matrix \mathbf{Q} we have to insert an indel in the alignment. The alignment corresponds to the score $(-3, -1)$ is as follows:

A	-	C	T	A	G	G	G
A	G	G	G	C	C	T	G

The time and space-complexity for each k value is similar to that of the DP method in

Section 2.3.3, page 16, that is, $O(n_1 \cdot n_2 \cdot (n_1 + n_2))$. Overall, the ϵ -constraint approach has $O(n_1 \cdot n_2 \cdot (n_1 + n_2))$ time and space-complexity, which are of similar order to the DP approach for the same problem (see Section 4.4.1). Note that, if the purpose is only to find the non-dominated score set, by keeping the matrix \mathbf{P} and \mathbf{Q} for each k and $k - 1$, we only require $O(n_1 \cdot n_2)$ -space to solve the problem.

Table 4.3: Matrices \mathbf{P} and \mathbf{Q} for $k = 0$ and $k = 1$ for the example in Section 4.4.1 with the ϵ -constraint approach. The numbers in the blue boxes show the traceback for the alignment with score $(-3, -1)$

$\mathbf{Q}[i, j, 0]:$									$\mathbf{Q}[i, j, 1]:$								
		A	C	T	A	G	G	G			A	C	T	A	G	G	G
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0	$-\infty$	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
A 1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	A 1	0	$-\infty$	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
G 2	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G 2	$-\infty$	1	$-\infty$	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
G 3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G 3	$-\infty$	$-\infty$	0	$-\infty$	-1	$-\infty$	$-\infty$	$-\infty$
G 4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G 4	$-\infty$	$-\infty$	$-\infty$	-1	$-\infty$	-2	$-\infty$	$-\infty$
C 5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	C 5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-2	$-\infty$	-3	$-\infty$
C 6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	C 6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-3	$-\infty$	-4
T 7	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	T 7	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-4	$-\infty$
G 8	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G 8	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-5

$\mathbf{P}[i, j, 0]:$									$\mathbf{P}[i, j, 1]:$								
		A	C	T	A	G	G	G			A	C	T	A	G	G	G
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0	$-\infty$	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
A 1	$-\infty$	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	A 1	0	$-\infty$	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
G 2	$-\infty$	$-\infty$	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G 2	$-\infty$	-1	$-\infty$	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
G 3	$-\infty$	$-\infty$	$-\infty$	-1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G 3	$-\infty$	$-\infty$	0	$-\infty$	-1	$-\infty$	$-\infty$	$-\infty$
G 4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-2	$-\infty$	$-\infty$	$-\infty$	G 4	$-\infty$	$-\infty$	$-\infty$	-1	$-\infty$	0	$-\infty$	$-\infty$
C 5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-3	$-\infty$	$-\infty$	C 5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-2	$-\infty$	-1	$-\infty$
C 6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-4	$-\infty$	C 6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-3	$-\infty$	-1
T 7	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-5	T 7	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-4	$-\infty$
G 8	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G 8	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-3

ϵ -constraint for Problem (VSGP)

To solve the sequence alignment problem with maximum substitution score and a fixed number of gaps, we use the extension of the Sankoff algorithm proposed in (Sankoff, 1972). This

DP algorithm finds the longest common subsequence (equivalent to maximize the number of the matches) with an inequality constraint on the number of gaps (except the gaps that arise in the start or at the end of the alignment). Here, we modified this algorithm to find an alignment with maximum substitution score, and equality constraint on the number of gaps.

The number of gaps in an alignment is bounded by the number of indels. Let $A = (a_1, \dots, a_{n_1})$ and $B = (b_1, \dots, b_{n_2})$ be two sequences and let ℓ_{cs} be the longest common subsequence between the two sequences. Then, $d_{\max} = n_1 + n_2 - 2\ell_{cs}$ is the maximal number of gaps in the alignment between the two strings.

For solving Problem (VSGP), we keep three matrices: \mathbf{P} , \mathbf{Q} and \mathbf{R} . Likewise the previous method, for a given $(i, j, k) \in \{1, \dots, n_1\} \times \{1, \dots, n_2\} \times \{0, \dots, d\}$, each entry $\mathbf{P}[i, j, k]$, $\mathbf{Q}[i, j, k]$ and $\mathbf{R}[i, j, k]$ stores the maximum score of aligning two subsequences (a_1, \dots, a_i) and (b_1, \dots, b_j) constrained to k gaps that end with (a_i, b_j) , $(a_i, '-')$ and $('-', b_j)$, respectively. The reason to use an extra matrix, \mathbf{R} , is to keep the track of gaps in both sequences in the alignments, i.e. matrix \mathbf{Q} keeps track of gaps in the first sequence whereas matrix \mathbf{R} keeps track of gaps in the second sequence. The recursion for $1 \leq i \leq n_1$, $1 \leq j \leq n_2$ and $0 < k \leq d_{\max}$ is as follows:

$$\mathbf{P}[i, j, k] = \max \begin{cases} \mathbf{P}[i-1, j-1, k] + M[a_i, b_j] \\ \mathbf{Q}[i-1, j-1, k] + M[a_i, b_j] \\ \mathbf{R}[i-1, j-1, k] + M[a_i, b_j] \end{cases} \quad (4.13)$$

$$\mathbf{Q}[i, j, k] = \max \begin{cases} \mathbf{P}[i, j-1, k-1] \\ \mathbf{Q}[i, j-1, k] \\ \mathbf{R}[i, j-1, k-1] \end{cases} \quad (4.14)$$

$$\mathbf{R}[i, j, k] = \max \begin{cases} \mathbf{P}[i-1, j, k-1] \\ \mathbf{Q}[i-1, j, k-1] \\ \mathbf{R}[i-1, j, k] \end{cases} \quad (4.15)$$

where M is a substitution matrix. The basis cases of the matrices for $0 \leq k \leq d$, $0 \leq i \leq n_1$ and $0 \leq j \leq n_2$ are:

$$\mathbf{Q}[i, j, 0] = \mathbf{R}[i, j, 0] = -\infty \quad (4.16)$$

$$\mathbf{P}[i, j, 0] = \begin{cases} 0 & \text{if } i = j = 0 \\ \mathbf{P}[i - 1, j - 1, 0] + M[a_i, b_j] & \text{if } i = j \neq 0 \\ -\infty & \text{if } i \neq j \end{cases} \quad (4.17)$$

$$\mathbf{P}[i, 0, k] = \mathbf{P}[0, j, k] = -\infty \quad k > 0 \quad (4.18)$$

$$\mathbf{Q}[i, 0, k] = \mathbf{R}[0, j, k] = -\infty \quad k > 0 \quad (4.19)$$

$$\mathbf{Q}[0, j, k] = \begin{cases} 0 & \text{if } k = 1 \\ -\infty & \text{if } k > 1 \end{cases} \quad (4.20)$$

$$\mathbf{R}[i, 0, k] = \begin{cases} 0 & \text{if } k = 1 \\ -\infty & \text{if } k > 1 \end{cases} \quad (4.21)$$

For the sequences in the example of Section 4.4.1, page 82, Table 4.4 shows the matrices \mathbf{P} , \mathbf{Q} and \mathbf{R} for $k = 1$ and $k = 2$. For the basis cases, $k = 0$ (when no indel/gaps is allowed), the matrix \mathbf{P} is exactly equal to that of Table 4.3 and matrices \mathbf{Q} and \mathbf{R} do not have feasible values. When $k = 1$, the maximum substitution score is -3 , which is the maximum between $\mathbf{P}[8, 7, 1] = -3$, $\mathbf{Q}[8, 7, 0] = -\infty$ and $\mathbf{R}[8, 7, 0] = -5$. In this case, the alignment corresponding to the score $(-3, -1)$ is the same as for Problem VSDP in Table 4.4. With $k = 2$, the maximum substitution score is 4, which corresponds to two different alignments, one from matrix \mathbf{P} , and other from matrix \mathbf{R} . The numbers in boxes show the traceback path to the corresponding Pareto optimal alignment with the score $(4, -2)$ from matrix \mathbf{R} . The alignment is as follows:

A	C	T	A	G	G	G	-	-	-	-
-	-	-	A	G	G	G	C	C	T	G

The removal of dominated scores is performed the same as the ϵ -constraint approach for Problem (VSDP). Since the number of gaps is bounded from above by the number of indels in alignment, the time and space complexity is also the same.

Table 4.4: ϵ -constraint matrices for Problem (VSGP) with $k = 1$ and $k = 2$ for the example in Section 4.4.1

		$Q[i, j, 1]$										$Q[i, j, 2]$							
		0	A	C	T	A	G	G	G			0	A	C	T	A	G	G	G
		0	1	2	3	4	5	6	7			0	1	2	3	4	5	6	7
0		$-\infty$	0	0	0	0	0	0	0	0		$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
A	1	$-\infty$	$-\infty$	1	1	1	1	1	1	A	1	$-\infty$	0	0	0	0	1	1	1
G	2	$-\infty$	$-\infty$	$-\infty$	0	0	0	0	0	G	2	$-\infty$	0	1	1	1	1	2	2
G	3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-1	-1	-1	-1	G	3	$-\infty$	0	1	1	1	1	1	3
G	4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-2	-2	-2	G	4	$-\infty$	0	1	1	1	1	1	2
C	5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-3	-3	C	5	$-\infty$	0	1	2	2	2	2	2
C	6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-4	C	6	$-\infty$	0	1	2	2	2	2	2
T	7	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	T	7	$-\infty$	0	1	1	3	3	3	3
G	8	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G	8	$-\infty$	0	1	1	1	2	2	2

		$P[i, j, 1]$										$P[i, j, 2]$							
		0	A	C	T	A	G	G	G			0	A	C	T	A	G	G	G
		0	1	2	3	4	5	6	7			0	1	2	3	4	5	6	7
0		$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0		$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
A	1	$-\infty$	$-\infty$	-1	-1	1	-1	-1	-1	A	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
G	2	$-\infty$	-1	$-\infty$	0	0	2	2	2	G	2	$-\infty$	$-\infty$	-1	-1	-1	1	2	2
G	3	$-\infty$	-1	0	$-\infty$	-1	1	3	3	G	3	$-\infty$	$-\infty$	-1	0	0	2	2	3
G	4	$-\infty$	-1	0	-1	$-\infty$	0	2	4	G	4	$-\infty$	$-\infty$	-1	0	0	2	3	3
C	5	$-\infty$	-1	2	-1	-2	$-\infty$	-1	1	C	5	$-\infty$	$-\infty$	1	0	0	0	1	2
C	6	$-\infty$	-1	2	1	-2	-3	$-\infty$	-2	C	6	$-\infty$	$-\infty$	1	0	1	1	1	2
T	7	$-\infty$	-1	0	3	0	-3	-4	$-\infty$	T	7	$-\infty$	$-\infty$	-1	3	1	1	1	2
G	8	$-\infty$	-1	0	-1	2	1	-2	-3	G	8	$-\infty$	$-\infty$	-1	1	2	4	4	4

		$R[i, j, 1]$										$R[i, j, 2]$							
		0	A	C	T	A	G	G	G			0	A	C	T	A	G	G	G
		0	1	2	3	4	5	6	7			0	1	2	3	4	5	6	7
0		$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0		$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
A	1	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	A	1	$-\infty$	0	0	0	0	0	0	0
G	2	0	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G	2	$-\infty$	0	1	1	1	1	1	1
G	3	0	1	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G	3	$-\infty$	0	1	1	1	2	2	2
G	4	0	1	0	-1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	G	4	$-\infty$	0	1	1	1	2	3	3
C	5	0	1	0	-1	-2	$-\infty$	$-\infty$	$-\infty$	C	5	$-\infty$	0	1	1	1	2	3	4
C	6	0	1	0	-1	-2	-3	$-\infty$	$-\infty$	C	6	$-\infty$	0	2	1	1	2	3	4
T	7	0	1	0	-1	-2	-3	-4	$-\infty$	T	7	$-\infty$	0	2	1	1	2	3	4
G	8	0	1	0	-1	-2	-3	-4	-5	G	8	$-\infty$	0	2	3	1	2	3	4

4.5 Experimental Analysis

4.5.1 Performance of the Pruning Technique

In this section, we evaluate the performance of the pruning technique on the DP approach described in the Section 4.4.1. We compare the DP with and without the pruning technique (DP-Prune and DP-noPrune, respectively) for both problem variants (VSDP) and (VSGP). The implementations were coded in C and compiled with gcc version 4.6.1 with the `-O3` compiler option, in a computer with two processors Intel Xeon 5620, 2.4 GHz, four core and 16 GB RAM, with operating system Ubuntu 11.10. Except for the compiler option, no other code optimization technique was used in the experiments.

We considered the sequences available from the benchmark BAliBase version 3.0 (Thompson et al., 2005) reference set 9. The subsets RV911, RV912, and RV913 were chosen since they are organized into three different groups according to the sequence variability: less than 20%, 20-40% and 40-80%, respectively. From the data sets we extracted the sequences from the following groups: RV911-BOX096 (12 sequences), RV911-BOX115 (7 sequences), RV911-BOX010 (18 sequences), RV912-BOX075 (13 sequences), RV912-BOX258 (16 sequences), RV912-BOX154 (5 sequences), RV913-BOX158 (55 sequences), RV913-BOX222 (7 sequences) and RV913-BOX063 (8 sequences). Our implementations were run on all pairs of sequences of the same group; the substitution matrix PAM250 (Dayhoff and Schwartz, 1978) was considered in our experiments.

Preliminary experiments indicated that only three bounds (MAX, MID and MIN) were insufficient for obtaining good performance in terms of running time. For this reason, several weighted sum problems were solved for different weight combinations in order to obtain a tighter lower bound set: 5, 10, 15 and 20. For each number w of combinations, the weights were varying in the following manner: $w_s = i$, $w_d = w - i$, $i = 1, \dots, w - 1$. Other experiments that we performed indicated that no improvement in terms of pruning could be obtained for w values larger than 20. In a second set of experiments for Problem (VSGP), we observed that the pruning technique was only being effective for entries $\mathbf{Q}[i, j]$, $i \geq n_1/2$ and $j \geq n_2/2$. Therefore, in order to reduce the overall CPU-time, we switched off the pruning technique for smaller indices.

Table 4.5: Experimental results for the pruning technique for Problem (VSDP)

Data sets	size	# <i>nd</i>	CPU-time (% <i>prun</i>)					
			DP-noPrune	DP-Prune (1)	DP-Prune (5)	DP-Prune (10)	DP-Prune (15)	DP-Prune (20)
RV911-BOX115	750	190	0.5 ± 0.2	0.4 ± 0.2 (45%)	0.4 ± 0.2 (53%)	0.4 ± 0.2 (58%)	0.4 ± 0.2 (59%)	0.4 ± 0.2 (60%)
RV911-BOX096	795	189	0.6 ± 0.3	0.5 ± 0.2 (38%)	0.4 ± 0.2 (55%)	0.4 ± 0.2 (60%)	0.4 ± 0.2 (62%)	0.4 ± 0.2 (63%)
RV911-BOX010	1134	269	2.1 ± 1.3	1.5 ± 1.2 (31%)	1.4 ± 1.1 (35%)	1.3 ± 1.1 (62%)	1.4 ± 1.2 (63%)	1.4 ± 1.2 (65%)
RV912-BOX075	457	114	0.1 ± 0.0	0.1 ± 0.0 (53%)	0.1 ± 0.0 (71%)	0.1 ± 0.0 (75%)	0.1 ± 0.0 (76%)	0.1 ± 0.0 (77%)
RV912-BOX258	607	106	0.3 ± 0.1	0.1 ± 0.1 (67%)	0.1 ± 0.1 (84%)	0.1 ± 0.1 (86%)	0.1 ± 0.1 (87%)	0.1 ± 0.1 (87%)
RV912-BOX154	1076	164	0.8 ± 0.4	0.5 ± 0.3 (59%)	0.4 ± 0.2 (71%)	0.4 ± 0.2 (74%)	0.4 ± 0.2 (75%)	0.4 ± 0.2 (76%)
RV913-BOX158	664	74	0.2 ± 0.1	0.1 ± 0.0 (83%)	0.0 ± 0.0 (91%)	0.0 ± 0.0 (92%)	0.0 ± 0.0 (92%)	0.0 ± 0.0 (92%)
RV913-BOX222	983	149	0.8 ± 0.2	0.3 ± 0.2 (72%)	0.2 ± 0.1 (86%)	0.2 ± 0.1 (88%)	0.2 ± 0.1 (89%)	0.2 ± 0.1 (89%)
RV913-BOX063	1374	206	2.0 ± 0.4	0.7 ± 0.3 (76%)	0.4 ± 0.2 (88%)	0.4 ± 0.2 (90%)	0.4 ± 0.2 (91%)	0.4 ± 0.2 (91%)

Table 4.6: Experimental results for the pruning technique for Problem (VSGP)

Data sets	size	# <i>nd</i>	CPU-time (% <i>prun</i>)					
			DP-noPrune	DP-Prune (1)	DP-Prune (5)	DP-Prune (10)	DP-Prune (15)	DP-Prune (20)
RV911-BOX115	750	239	3.8 ± 0.9	3.5 ± 1.4 (1%)	3.3 ± 1.4 (10%)	3.3 ± 1.4 (16%)	3.4 ± 1.5 (19%)	3.5 ± 1.5 (21%)
RV911-BOX096	793	264	4.9 ± 2.2	4.6 ± 3.5 (0%)	4.5 ± 3.6 (9%)	4.5 ± 3.7 (15%)	4.6 ± 3.8 (18%)	4.8 ± 3.9 (19%)
RV911-BOX010	1133	382	13.7 ± 5.6	13.6 ± 9.1 (0%)	12.8 ± 8.8 (10%)	12.6 ± 8.9 (14%)	12.8 ± 9.1 (18%)	13.1 ± 3.4 (22%)
RV912-BOX075	458	139	0.8 ± 0.1	0.8 ± 0.1 (1%)	0.6 ± 0.1 (30%)	0.5 ± 0.1 (46%)	0.5 ± 0.1 (52%)	0.5 ± 0.1 (54%)
RV912-BOX258	608	153	1.8 ± 0.5	1.8 ± 0.8 (3%)	1.2 ± 0.5 (43%)	0.9 ± 0.4 (59%)	0.8 ± 0.4 (65%)	0.8 ± 0.4 (69%)
RV912-BOX154	1076	243	5.3 ± 1.2	5.3 ± 2.0 (1%)	4.7 ± 1.6 (14%)	4.3 ± 1.4 (26%)	4.2 ± 1.3 (31%)	4.2 ± 1.3 (34%)
RV913-BOX158	671	69	1.1 ± 0.2	1.3 ± 0.3 (2%)	0.5 ± 0.2 (66%)	0.3 ± 0.1 (79%)	0.3 ± 0.1 (82%)	0.3 ± 0.1 (83%)
RV913-BOX222	983	180	5.1 ± 0.8	4.9 ± 1.3 (5%)	2.7 ± 0.7 (47%)	1.9 ± 0.6 (65%)	1.7 ± 0.6 (71%)	1.6 ± 0.6 (74%)
RV913-BOX063	1374	275	10.4 ± 1.5	10.6 ± 2.6 (5%)	7.3 ± 2.1 (54%)	5.0 ± 1.7 (70%)	4.3 ± 1.5 (75%)	4.1 ± 1.4 (77%)

Tables 4.5 and 4.6 give the results obtained for both problem variants, where column *size* is the average sequence length, *#nd* corresponds to the average number of non-dominated states, *CPU-time* gives the average and standard deviation of CPU-times in seconds to terminate and *%prun* gives the percentage of states that were pruned in the DP-Prune(w) version. A bold value indicates the best average CPU-time; in case of a tie, the value with the largest pruning percentage was chosen, since it suggests less memory usage.

The results show that the DP-Prune version is able to prune from 31% to 92% of the states that are generated by the DP-noPrune version for Problem (VSDP). The improvement in terms of CPU-time can go up to 80% in the set RV913-BOX063. However, no improvement can be found for RV912-BOX075 and RV912-BOX154, although all versions were extremely fast in those cases (less than 0.5 seconds). For Problem (VSGP), the pruning can reach 83% and CPU-time improved up to 60% in RV913-BOX222. In both problems, it is possible to observe that the increase of parameter w does not translate directly into faster CPU-time; for instance, in the sets RV911 for Problem (VSGP), the best CPU-time was obtained with $w = 10$, although better pruning percentage was obtained with $w = 20$ ($\geq 20\%$). Clearly, the larger, the lower bound set, the higher the required time for comparison. Moreover, the pruning seems to be more effective for large levels of residue identity.

It is also noteworthy to mention that the number of non-dominated states is a small fraction of the average size of the genes. We also observed that both algorithms on Problem (VSGP) take roughly 4 to 5 times more CPU-time than on Problem (VSDP).

4.5.2 Computational Analysis with Real Data

In this section, we report computational analysis on real data set for comparing the performance of the DP algorithms and ϵ -constraint approaches for both problem variants (VSDP) and (VSGP) (see Sections 4.4.1 and 4.4.2). Moreover, we also examine the effectiveness of the pruning technique described in Section 4.4.1, page 76.

Two real data set are considered. The first data set consists of *Hepatocystis sp. CNRP11*, *Hepatocystis sp. PP1*, *Plasmodium juxtannucleare*, *Breviata anathema* (AF153206), *Theileria parva* (AAGK01000006), *Parvilucifera infectans* (AF133909), *Amylax triacantha* (AB375869), *Ceratocorys horrida* (AF022154) and *Alexandrium catenella* (AB088280). The second data set consists of *Candida* genes, *C.albicans PAPa*, *C.albicans PAPalpha*, *C.tropicalis PAPalpha*, *C.tropicalis PAPa* and *C.dublinsiensis PAPa*, as well as the genes *Pichia stipitis PAPalpha* and *Saccharomyces cerevisiae PAP* (See Butler et al. (2009) for a more detailed description of these genes).

Table 4.7: Experimental results on the two data sets

Variant	Set	# <i>nd</i>	% <i>prun</i>	% <i>impr</i>	CPU-time		
					DP-Prune	DP-noPrune	EC
(VSDP)	1	348	57.7%	57.8%	0.8 ± 0.1	1.9 ± 0.3	8.4 ± 1.8
	2	401	46.2%	60.0%	0.9 ± 0.3	2.3 ± 0.8	10.8 ± 1.6
(VSGP)	1	302	41.0%	39.6%	7.9 ± 1.1	13.1 ± 2.4	23.4 ± 4.5
	2	397	34.0%	31.2%	10.1 ± 1.9	14.7 ± 3.2	22.5 ± 3.8

The average size of the genes in the first and second data set is 1724 and 1703, respectively. The implementations were coded in C and compiled with gcc version 4.4.3 with the -O3 compiler option, in a cluster with 16 nodes. Each node contains Intel Core i7 CPU with 4-core and 2 GB RAM, with operating system Ubuntu 10.04.4. Except for the compiler option, no other code optimization technique was used in the experiments.

Preliminary experiments with DP using the pruning technique indicated that three bounds were also insufficient for obtaining good performance for Problem (VSDP). For this reason, several weighted sum problems were solved for different weight combinations in order to obtain a tighter lower bound set. A reasonable good trade-off between the time spent on the pre-processing phase and the time spent on comparing bounds was achieved with the number of bounds equal to 24, with weights varying in the following manner: $w_s = i$, $w_g = 25 - i$, $i = 1, \dots, 24$.

Table 4.7 gives the results obtained for problem variants (VSDP) and (VSGP) for both data sets. The DP algorithm with pruning and without pruning as well as the ϵ -constraint technique are shown with DP-Prune, DP-noPrune and EC, respectively. Column #*nd* corresponds to the average number of non-dominated states, %*prun* gives the percentage of states that were pruned in the DP-Prune version, CPU-time gives the average and standard deviation of CPU-time in seconds to terminate and %*impr* corresponds to the percentage of improvement of DP-Prune in terms of CPU-time in comparing to DP-noPrune version. The values are averaged over all pairs of genes.

The results show that EC has the worst performance in terms of CPU time. We observed that on Problem (VSDP) it takes nearly 10 times more than DP-Prune and DP-noPrune and on Problem (VSGP) take almost twice more from DP-Prune and DP-noPrune version. Moreover, the DP-Prune version is able to prune between 34% and 57% of the states that are generated by the DP-noPrune version. This has a direct consequence on CPU-time; the DP-Prune version is between 31% and 60% faster. It is noteworthy to mention that the

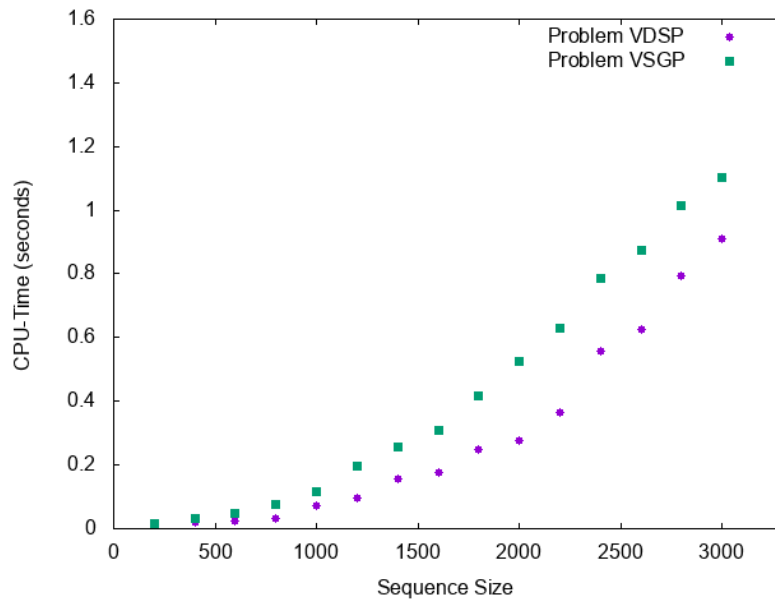


Figure 4.3: The experimental results for DP-Prune for Problems (VSDP) and (VSGP) with random data set ranged from $n = 200$ to 3000

number of non-dominated states is close to $1/5$ of the average size of the genes. We also observed that both algorithms on Problem (VSGP) take roughly 7-10 times more CPU-time than on Problem (VSDP).

4.5.3 Computational Analysis with Random Data

In the previous section, the experimental results suggested that the DP algorithm using the pruning technique (DP-Prune) has the best performance, whereas ϵ -constraint (EC) takes much more time. In this section, we analysed the performance of DP-Prune on various data length for both problem variants (VSDP) and (VSGP). The implementations were run on a wide range of randomly generated DNA sequences. The size of sequences ranged from $n = 200$ to 3000. A total of 30 instances were generated for each combination of values of n . We used the same system to run the experiments as in Section 4.5.2.

Figure 4.3 presents the experimental results obtained for Problem (VSDP) and (VSGP). The values are averaged over 30 instances of the same size.

²The randomly sequences were obtained from http://www.bioinformatics.org/sms2/random_dna.html Stothard (2000).

4.6 Summary and Discussion

In this chapter, we introduced the multiobjective pairwise sequence alignment problem and proposed the extensions of dynamic programming algorithms for several problem variants. Furthermore, a pruning technique is presented to improve the performance of dynamic programming algorithms for multiobjective pairwise sequence alignment, which uses lower and upper bounds to discard states in the early stages of the process. This technique can easily be extended to the case of the affine gap by performing the necessary changes in the recurrence relation of matrices \mathbf{S} and \mathbf{T} (see Section 4.4.1). They can also be extended to the three criteria case, where the substitution score, the number of indels and the number of gaps are simultaneously considered in the score vector function.

In the next chapter, the other main contributions of this thesis are introduced, in particular, the multiobjective formulation of the multiple sequence alignment and algorithms to solve it.

Chapter 5

Multiobjective Multiple Sequence Alignment

5.1 Introduction

In this chapter, we consider alignment problems with more than two sequences, called the *Multiple Sequence Alignment* (MSA) problem. These alignments play a crucial role in molecular biology. Many algorithms have been proposed for the inference of multiple sequence alignments of protein and DNA sequences (Carrillo and Lipman, 1988).

The single-objective version of the MSA problem for an arbitrary number of sequences is known to be NP-hard (Just, 2001). For this reason, most of the current approaches to this problem are based on heuristics, ranging from progressive to iterative methods. In principle, a multiobjective variant of the MSA is even harder from the computational point of view since more than one solution is needed to attain. Therefore, in order to obtain solutions in a short amount of time, we need to resort to heuristics for multiobjective optimization.

There have been several variants of this problem. In our particular case, we are considering the MSA problem with sum-of-pairs or weighted sum-of-pairs score (see Section 2.4.1, page 24, for more details). There is a consensus that these scores reflect biological events and that the optimization of them leads to biologically correct alignments. However, these functions are highly dependent on the parameter setting, like the relative importance of indels/gaps with respect to the increase of the substitution score. For the same reasons as

presented for the PSA problem (see Section 4.1, page 71), we propose a new multiobjective formulation for the sum-of-pairs score.

The main contributions of this chapter are explained in the following, which were partially published in a journal paper (Abbasi et al., 2016), a conference paper (Abbasi et al., 2015) and a poster with an oral presentation in an international conference (Abbasi et al., 2013b).

- i) We propose to approach the *Multiobjective Multiple Sequence Alignment* (mMSA) problem with a local search algorithm, extending the formulation given in Section 4.2, page 72, for an arbitrary number of sequences. This approach considers the sum-of-pairs score function that maximizes the substitution score, based on a given substitution matrix, and minimizes the number of indels/gaps. The local search algorithm starts from alignments obtained from some well-known single-objective approaches, such as Clustal Omega and T-Coffee, as well as from randomly generated alignments.
- ii) We propose an iterated local search algorithm to improve the quality of the alignments produced by the local search algorithm. This procedure consists of successive local search runs, each of which starting from solutions obtained from previous runs that were slightly perturbed.
- iii) We present a modification of NSGA-II (Deb et al., 2002) to solve the mMSA. In the literature, NSGA-II has been the main approach to solve the several variants of this problem. In particular, we adapt the NSGA-II proposed by MOSAStrE, a known approach for a different variant of the multiobjective sequence alignment (Ortuño et al., 2013). The same type of genetic operators are applied.
- iv) The heuristic approaches proposed in this study are tested thoroughly under different parameter options such as starting alignments, neighborhood definitions, and perturbation mechanisms. In the experimental analysis, we used all the instances obtained from the benchmark database BALiBASE 3.0, subsets RV11, RV12 and RV20 (Thompson et al., 2005). We used the hypervolume indicator to evaluate the performance of the algorithms. Moreover, in order to assess local search and NSGA-II, we also applied the attainment function methodology to visualize differences of performance in the objective space.

This chapter is outlined as follows: Section 5.2 provides the required notation and definitions for the mMSA problem that is tackled in this thesis; Section 5.3 presents a review of

known heuristic approaches for several different variants of mMSA; Section 5.4 describes the proposed local search algorithm for our variant of the mMSA problem as well as implementation details of NSGA-II for this problem; Section 5.5 describes an in-depth experimental assessment of the proposed methods and, finally, Section 5.6 presents the conclusions and topics for future work.

5.2 Notation and Definitions

A number of different objectives have been formulated in the context of mMSA Problem, such as TC (percentage of aligned columns), NonGaps (percentage of columns that do not contain indels), SP-score (sum-of-pairs score), STRIKE, Entropy, BAliScore and MetAl (see Section 2.4.3, page 37, for more details). Among most of them, the sum-of-pairs score is the most used objective. However, as mentioned in Chapter 4, this score is naturally multiobjective.

In this section, we introduce a particular definition and formulation of the mMSA problem. Analogous to the PSA problem, we disaggregate the two objective components of the sum-of-pairs score: the substitution score and indels/gaps score.

Definition 5.2.1. *Let $A_1 = (a_{11}, \dots, a_{1n_1}), \dots, A_m = (a_{m1}, \dots, a_{mn_m})$ be m sequences over an alphabet Σ . Let $'-'$ $\notin \Sigma$ be an indel character and let $\Sigma' = \Sigma \cup \{'-\'}$. Let ϕ be an alignment of m sequences A_1, \dots, A_m (see Definition 2.4.1, page 25). Consider ℓ to be the length of the multiple sequence alignment. We define the following biobjective sum-of-pairs (BSP) score of an alignment.*

$$BSP(\phi) = (BSP_s(\phi), BSP_d(\phi))$$

where

$$BSP_s(\phi) = \sum_{j=1}^{\ell} \sum_{i=1}^{m-1} \sum_{k=i+1}^m s(a_{ij}, a_{kj})$$

$$BSP_d(\phi) = \sum_{j=1}^{\ell} \sum_{i=1}^{m-1} \sum_{k=i+1}^m d(a_{ij}, a_{kj})$$

The score $s(a_{ij}, a_{kj})$, for $a_{ij}, a_{kj} \in \Sigma$, is obtained from a substitution matrix, if neither $a_{ij} = '-'$ nor $a_{kj} = '-'$, otherwise is 0. The score $d(a_{ij}, a_{kj})$, for $a_{ij}, a_{kj} \in \Sigma'$ is 1, if either $a_{ij} = '-'$ or $a_{kj} = '-'$, otherwise is 0.

Let Φ denote the set of all feasible alignments. The mMSA problem consists of finding the alignments that are “maximal” with respect to the above score function. By considering the score function presented in Definition 5.2.1, we define a biobjective multiple sequence alignment problem as follows:

$$\arg \text{vmax} \{ \phi : BSP(\phi), \phi \in \Phi \}$$

where $\arg \text{vmax}$ is understood in terms of Pareto optimality. The image of set of Φ in the score function space is called *feasible score set*. The notion of Pareto optimality is described as follows: Given two feasible alignments ϕ and ϕ' , $BSP(\phi) \succ BSP(\phi')$ (ϕ dominates ϕ') if and only if it holds that $BSP_s(\phi) \geq BSP_s(\phi')$ and $BSP_d(\phi) \leq BSP_d(\phi')$, with at least one strict inequality. An alignment ϕ^* is *Pareto optimal* if there exists no other alignment ϕ such that $BSP(\phi) \succ BSP(\phi^*)$. In this section we use the same terminology as used in Section 3.2.1, page 45, that is, the set of all Pareto optimal alignments is called *Pareto optimal alignment set*. The image of a Pareto optimal alignment in the score space is a *non-dominated score* and the set of all non-dominated scores is called *non-dominated score set*.

Although the multiobjective pairwise sequence alignment problem can be solved efficiently, that is, the running time to find the non-dominated score set is a polynomial function of the size of the sequences, this is no longer the case for the multiple counterparts for an arbitrary number of sequences. Thus, the goal, in practice, is to find an approximation to the non-dominated score set in a reasonable amount of time, which can be performed by heuristics methods (see Section 3.3.2, page 51). In the following section, we review some of the mMSA problem variants and the proposed methods to tackle them.

5.3 Review of Approaches to the mMSA Problem

In this section, we review several formulations of the mMSA problem that can be found in the literature. These formulations are considerably different from ours and among them, which make it difficult to approach this problem in a more general manner.

One of the first known approaches is MOSAStrE, Multiobjective Optimizer for Sequence Alignments based on Structural Evaluations, proposed by [Ortuño et al. \(2013\)](#). The authors considered a multiobjective MSA problem with the following objectives: i) STRIKE score, score with structural information (see Section 2.4.3, page 40), ii) Total Column (TC) (the percentage of identical aligned columns in the alignment) and iii) Percentage of non-gaps

parts (the percentage of columns in the alignment that contains indels). They used a genetic algorithm to solve this problem that is based on NSGA-II with a single-point crossover operator and mutation based on shifting a gap in the alignment. On their experiments, the initial alignments are obtained from 8 (single-objective) MSA programs: ClustalW (Thompson et al., 1994), Muscle (Edgar, 2004), Kalign (Lassmann and Sonnhammer, 2005), Mafft (Katoh et al., 2002), (Szabó et al., 2010), T-Coffee (Notredame et al., 2000), Fast statistical alignment (FSA) (Bradley et al., 2009) and ProbCons (Do et al., 2005).

Soto and Becerra (2014) considered two objectives: Entropy and MetAl (see Section 2.4.3, page 38, for more information). Similar to Ortuño et al. (2013) they used a genetic algorithm called, MOEA, also inspired by NSGA-II. A two-point crossover and a random shift of an indel are used as crossover and mutation methods, respectively. They have selected six algorithms to provide starting alignments, namely ClustalW, Muscle, MAFFT, ProbCons, T-Coffee and Clustal Omega (Sievers et al., 2011).

Kaya et al. (2014) considered three objectives: similarity, affine gap penalty and support (see Section 2.4.3, page 39). They proposed an algorithm called MSAMGA to solve this problem, which is also based on NSGA-II. Two crossover operators (single and two-point) and three mutation operators, namely random changing and shifts toward the right and left of the gap were applied to a problem dataset from BALiBASE 2.0.

Zhu et al. (2016) proposed another multiobjective version that takes into account the SP-score and affine gap penalty. They introduced an approach called MOMSA, which is based on MOEA/D. The idea of MOEA/D is to convert a multiobjective optimization problem into a number of scalarized optimization problems (Zhang and Li, 2007). In MOMSA, the initial population is generated by adding randomly located indels on the alignment returned by ClustalW. The performance of this technique was tested over the benchmark datasets BALiBASE 2.0 and BALiBASE 3.0.

More recently, Zambrano-Vega et al. (2017) considered a three objective formulation of the MSA problem that includes the STRIKE score, the percentage of aligned columns and the percentage of non-gap symbols. They did an experimental study to compare the performance of four heuristic approaches to solve this problem variant. In particular, they applied NSGA-II (Deb et al., 2002), NSGA-III (Jain and Deb, 2014), GWASF-GA (Saborido et al., 2017), and MOCeLL (Nebro et al., 2009). The crossover and mutation operators, similar to MOSAStrE, are single-point crossover and shifting gap mutation, respectively. The method for filling the initial population is the same as used in MOSAStrE. Instead of creating random solutions, a number of pre-computed alignments are obtained from a set

of MSA programs. They considered large datasets of instances from BALiBASE 3.0 for the experimental analysis.

The methods presented in the above review formulate the MSA problem as a multiobjective optimization problem with different score definitions. The sum-of-pairs score is one of the key objectives that is used in Soto and Becerra (2014) and Zhu et al. (2016).

In this work, we formulate the MSA problem as an extension of the PSA problem (see Chapter 4) using substitution score and indels/gaps as different objectives to optimize. Since the problem has some combinatorial nature (assigning characters to positions in an alignment), we tackled it with local search approach, which has given good performance to combinatorial optimization problems (Hoos and Stützle, 2004). Moreover, given that most of the approaches above use NSGA-II, we also adapt it to our problem for comparison purpose.

5.4 Algorithms

In this section, we use the working principle of the Pareto Local Search (PLS), an iterative improvement search strategy, to find a set of alignments (see Section 3.3.2, page 62). Then, we describe the options that are considered to apply PLS to the mMSA problem: starting alignment and neighborhood function. We propose a k -block neighborhood to create neighboring alignments. Next, we improve PLS by using the same working principle of iterated local search. This procedure iterates over the PLS algorithm by perturbing some of the alignments in order to escape from local optima. Finally, we introduce NSGA-II (Deb et al., 2000, 2002) to solve the mMSA problem following the same working principle of MOSAStrE.

5.4.1 Pareto Local Search (PLS)

Pareto local search (Paquete et al., 2007) is a generic local search framework for multiobjective optimization problems. The algorithm starts with a feasible solution and searches *locally* for better neighbors to replace the current one. This neighborhood search is repeated until no improvement can be achieved, which means that the algorithm reached a local optimum. The pseudo code of Pareto Local Search with more details has been presented in Section 3.3.2, page 63. In the context of the mMSA problem, the neighborhood function associates every feasible alignment ϕ to a set of feasible alignments $N(\phi)$. An alignment ϕ is a Pareto local optimum if there exists no alignment ϕ' in $N(\phi)$ such that $BSP(\phi') \succ BSP(\phi)$. In

Table 5.1: Single objective methods that are used to create starting alignments

<p>i. Rand: Rand is a random possible alignment for comparison of the results with minimum possible number of indels.</p> <p>ii. Clustal Omega (Clust): A feasible alignment obtained from program Clustal Omega (Sievers et al., 2011) version 1.2.3 (available at http://www.clustal.org/omega).</p> <p>iii. T-Coffee: A feasible alignment obtained from program T-Coffee (Notredame et al., 2000) version 8.97 (available at http://www.tcoffee.org/Projects/tcoffee/).</p> <p>iv. MUSCLE: A feasible alignment obtained from program MUSCLE (Edgar, 2004) version 3.8.31(available at http://www.drive5.com/muscle/downloads.htm).</p> <p>v. Kalign: A feasible alignment obtained from program Kalign (Lassmann and Sonnhammer, 2005) version 2.04 (available at http://msa.sbc.su.se/cgi-bin/msa.cgi).</p> <p>vi. Maft: A feasible alignment obtained from program Maft (Katoh et al., 2002) version 7.305 (available at http://mafft.cbrc.jp/alignment/software/).</p> <p>vii. RetAlign: A feasible alignment obtained from program RetAlign (Szabó et al., 2010) version 1.0 (available at http://phylogeny-cafe.elte.hu/RetAlign/).</p> <p>viii. ProbCons: A feasible alignment obtained from program ProbCons (Do et al., 2005) version 1.12 (available at http://probcons.stanford.edu/download.html).</p> <p>ix. FSA: A feasible alignment obtained from program FSA (Bradley et al., 2009) version 1.15.5 (available at http://fsa.sourceforge.net/).</p>
--

the following, we describe the options that are considered to apply PLS the mMSA problem such as starting alignment and neighborhood function.

Representation of the Alignment

Alignments are represented as a $m \times n$ matrix, where m is the number of sequences, and n is the maximum length that alignment can be extended. Note that a multiple sequence alignment rarely contains indels/gaps more than 20% of the length of the largest sequence (Chelapilla and Fogel, 1999). Therefore, by considering the length of the largest sequence equal to ℓ_{max} , the alignment is bounded by a matrix of size $m \times 1.2\ell_{max}$.

Starting Alignment

The starting solution may have a substantial impact on the overall performance of local search algorithms. Therefore, we considered different possibilities for generating alignments by using different well-known single-objective methods. We follow the ideas introduced in [Ortuño et al. \(2013\)](#). We have generated, for every dataset, a number of alignments by using MUSCLE, Kalign, Mafft, RetAlign, T-Coffee, ProbCons, FSA and Clustal Omega. In addition, we consider a randomly generated alignment (Rand) for comparison purpose. This alignment is obtained by inserting indels randomly into the sequences, except in the largest one. We considered this initialization option to have a feasible alignment with the least possible number of indels. Table 5.1 lists all the methods applied to generate the starting alignments.

Neighborhood

We propose a *k-block* neighborhood technique for this problem. For a given sequence, its neighborhood is obtained by exchanging, at each gap, the adjacent left and right substrings of at most k characters with the indels in the *gap*. In the following, we establish the conditions for two alignments to be *k-block* neighbors.

Let $A_1 = (a_{1,1}, \dots, a_{1,n_1})$, ..., $A_m = (a_{m,1}, \dots, a_{m,n_m})$ denote m sequences and let ϕ_B and ϕ_C be two alignments, where $\phi_B = B_1, \dots, B_m$ and $\phi_C = C_1, \dots, C_m$. The alignments ϕ_B and ϕ_C are *k-block* neighbors if and only if the following conditions hold:

- i)* $|B_i| = |C_i|$, for $i = 1, \dots, m$;
- ii)* Let $J = \{j \mid j \in \{1, \dots, m\} : B_j \neq C_j\}$; then $|J| = 1$;
- iii)* For $j \in J$ let $\pi_{B_j}^i$ and $\pi_{C_j}^i$ denote the position of $a_{ji} \in A_j$ in sequence B_j and C_j , respectively. Let $I_{B_j} = \{\pi_{B_j}^i \mid \pi_{B_j}^i \neq \pi_{C_j}^i, i = 1, \dots, |A_j|\}$ and $I_{C_j} = \{\pi_{C_j}^i \mid \pi_{C_j}^i \neq \pi_{B_j}^i, i = 1, \dots, |A_j|\}$. Then, $\ell = |I_{B_j}| = |I_{C_j}| \leq k$.
- iv)* Let $I_{B_j} = \{I_{B_j,1}, \dots, I_{B_j,\ell}\}$ and $I_{C_j} = \{I_{C_j,1}, \dots, I_{C_j,\ell}\}$; then $I_{B_j,i+1} - I_{B_j,i} = I_{C_j,i+1} - I_{C_j,i} = 1$, for $i = 1, \dots, \ell - 1$.

Conditions *i)* and *ii)* state that both alignments should have the same size, and differ only in the j -th sequence, respectively. In addition, conditions *iii)* and *iv)* state that at most k characters from sequence A_j do not occupy the same position in both alignments, and that those characters are contiguous. For illustration purpose, consider the following

alignment. The shaded cells show the position of exchanged characters to create a new neighbor alignment.

D	-	G	G	F	-
-	D	-	F	G	L

The three possible 2-block neighbors for the first indel in the first sequence are as follows:

-	D	G	G	F	-
-	D	-	F	G	L

D	G	-	G	F	-
-	D	-	F	G	L

D	G	G	-	F	-
-	D	-	F	G	L

For the second indel in the first sequence, we obtain the following neighbors:

D	-	G	-	G	F
-	D	-	F	G	L

D	-	G	G	-	F
-	D	-	F	G	L

The only possible 2-block neighbor for the first indel in the second sequence is as follows:

D	-	G	G	F	-
D	-	-	F	G	L

For the second indel in the second sequence we obtain the following neighbors:

D	-	G	G	F	-
-	-	D	F	G	L

D	-	G	G	F	-
-	D	F	-	G	L

D	-	G	G	F	-
-	D	F	G	-	L

It is possible to visit all k -block neighbors of an alignment ϕ in a straightforward way. Given the first sequence of an alignment ϕ , consider the leftmost substring of size one that contains only a character and an indel to its right. Then, exchange that character with every indel in its right and stop when no indel is found. In case the substring has also indels to its left, repeat the same exchange procedure. If it is still possible, increase the size of the substring by one and repeat the same moves to its right and its left; repeat the overall procedure until reaching a subsequence of size k . Then, consider the next substring of size one to the right and repeat the same process as above. Note that each exchange generates a

Algorithm 8 Iterated Pareto Local Search for mMSA**procedure** IPLS

```

 $\Phi = \emptyset$ 
 $\Phi_0 = \text{GenerateInitialAlignments}()$ 
for each  $\phi \in \Phi_0$  do
     $\Phi' = \text{PLS}(\phi)$ 
     $\Phi = \text{Merge-NonDominated}(\Phi, \Phi')$ 
repeat
     $\Phi_P = \text{Selection}(\Phi)$ 
    for each ( $\phi_P \in \Phi_P$ ) do
         $\phi_{P'} = \text{Perturbation}(\phi_P)$ 
         $\Phi' = \text{PLS}(\phi_{P'})$ 
         $\Phi = \text{Merge-NonDominated}(\Phi, \Phi')$ 
until termination condition met
return  $\Phi$ 

```

new k -block neighbor. In order to maintain feasibility, the columns that contain only indels are deleted from the alignment.

5.4.2 Iterated Pareto Local Search (IPLS)

The PLS algorithm may get trapped in a set of Pareto local optima of poor quality. One possibility for escaping from Pareto local optima is to *perturb* one or more alignments in the archive and restart the PLS from those alignments (see Section 3.3.2, page 64). Iterated Pareto Local Search (IPLS) implements this search behavior. Its main steps are presented in Algorithm 8. Similar to the single-objective counterpart (see Lourenço et al. (2013)), four components have to be specified in IPLS:

- i) **GenerateInitialAlignments**: This generates an initial set of alignments Φ_0 ; based on the experimental results, we consider different starting alignments generated by single-objective methods such as T-Coffee, Clustal Omega, Muscle, Kalign, Mafft, RetAlign, ProbCons, FSA and Rand (see Table 5.1 for more details).
- ii) **Perturbation**: This procedure modifies some of the current alignments in set Φ leading to some intermediate set of alignments Φ' . In order to do that, first, p alignments are selected for perturbation from the set Φ . Based on preliminary experiments, we decided

to choose alignments in such a way that the corresponding non-dominated scores are spread enough in the objective space. This way, we avoid applying perturbation to two alignments that are, in principle, similar to each other. For the selection of p alignments for perturbation, we used a dynamic programming approach described in [Kuhn et al. \(2016\)](#)¹. Then, the alignments of the selected p points are perturbed and each of which is used as a starting solution for PLS, following a randomized ordering. For perturbing the alignments, we used the ideas applied in [Zhu et al. \(2016\)](#) to create an initial population for MOMSA, in which for each alignment, a gap of a given size N_{in} is inserted into each sequence of the alignment. The sequences are randomly split into two groups, and for each group, the gap is inserted in two randomly chosen positions. Finally, the two groups of sequences are merged to form a complete alignment.

- iii) **PLS**: This procedure is applied to an alignment and returns a set of non-dominated scores (see Algorithm 6, page 63).
- iv) **Merge-NonDominated**: This procedure merges two non-dominated sets and filters out the dominated scores.
- v) **Termination condition**: Different termination conditions can be applied such as i) algorithm stops by itself when it is not possible to find more non-dominated neighboring alignments, ii) a predefined time limit of CPU-time is reached or iii) after computing a predefined number of score vector function evaluations.

5.4.3 NSGA-II

NSGA-II ([Deb et al., 2000, 2002](#)) is a classic multiobjective evolutionary algorithm that generates new individuals from the original population by applying the standard genetic operators (selection, crossover, and mutation). A ranking procedure is applied to select the elements in the population, and a density estimator (the crowding distance) is used to diversify the selection (see Section 3.3.2, page 59, for more details). In the following, we illustrate some options that are adapted from MOSAStrE ([Ortuño et al., 2013](#)) to apply NSGA-II to our mMSA problem: solution representation, initial population (starting alignments) and genetic operator.

- i) **Solution representation**: The NSGA-II uses the solution representation described in Section 5.4.1, page 107.

¹The software is available at <https://eden.dei.uc.pt/~paquete/HSSP/>.

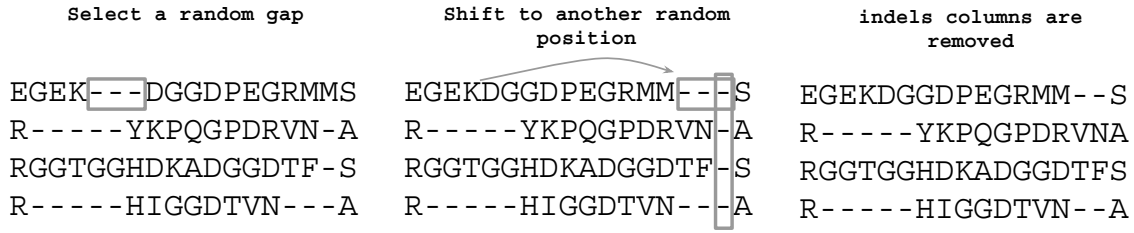


Figure 5.1: Mutation operator: gaps are randomly chosen and shifted to another position. Columns with only indels are removed

ii) **Initialize population:** A number of alignments are generated by using the single-objective methods from Table 5.1. For a population of size n , i alignments are generated and added to the initial population, and the remaining $n - i$ are created by applying the crossover operator to pairs of randomly selected alignments which are taken from the same initial population that is being created.

iii) **Genetic Operators:**

a) *Mutation:* This mutation operator is adapted directly from the MOSAStrE algorithm. In this operator, a set of random gaps is shifted to another random position in the same sequence. This operator is illustrated in Figure 5.1. In [Ortuño et al. \(2013\)](#), the authors consider this operator from two different aspects: first, new, but closely related, variants of the existent alignments are introduced into the population; secondly, columns containing only gaps are removed, which may produce alignments with less number of gaps.

b) *Crossover:* This operator is also adapted from the MOSAStrE algorithm. The crossover operator is a single-point crossover over alignments as proposed in [da Silva et al. \(2010\)](#). Figure 5.2 illustrates this operation. The procedure randomly selects two alignments (Figure 5.2 A). Then, on the first alignment, it selects a random position and splits it into two blocks. Afterward, for each sequence of the second alignment, it finds the position where the split has occurred in the first alignment. (Figure 5.2 B). In order to cross the blocks from both alignments, the blocks of the second alignment are filled with indels until the subsequences of the blocks of the second alignment have equal sizes (Figure 5.2 C). Finally, the blocks of two parents are crossed between them to create the child alignments (Figure 5.2 D).

A) Select two random alignment

Parent Alignment 1 (X)	Parent Alignment 2 (Y)
EGEK---DGGDPEGRMMS	EGEK-DGGD--PEGRMMS
R-----YKPQGPDRVNA	--R--YKPQ--GPDRVNA
RGGTGGHDKADGGDTFDS	RGGTGGHDKADGGDTFDS
R-----HIGGDTVNA	---R--HI---GGDTVNA

B) Cut random X in two blocks and tailor Y

X1	X2	Y1	Y2
EGEK---DGGDPEGRMMS		EGEK-DGGD--PEGRMMS	
R-----YKPQGPDRVNA		--R--YKPQ--GPDRVNA	
RGGTGGHDKADGGDTFDS		RGGTGGHDKADGGDTFDS	
R-----HIGGDTVNA		---R--HI---GGDTVNA	

C) Fill the blocks of Y with necessary indels

X1	X2	Y1	Y2
EGEK---DG	GDPEGRMMS	EGEK-DG--	-GD--PEGRMMS
R-----YK	PQGPDRVNA	--R--YK--	-PQ--GPDRVNA
RGGTGGHDK	ADGGDTFDS	RGGTGGHDK	---ADGGDTFDS
R-----	HIGGDTVNA	---R--HI-	HI---GGDTVNA

D) crossover the two alignment

Child Alignment 1 (Z1)	Child Alignment 2 (Z2)																				
<table> <thead> <tr> <th>X1</th> <th>Y2</th> </tr> </thead> <tbody> <tr> <td>EGEK---DG</td> <td>GD--PEGRMMS</td> </tr> <tr> <td>R-----YK</td> <td>PQ--GPDRVNA</td> </tr> <tr> <td>RGGTGGHDK</td> <td>--ADGGDTFDS</td> </tr> <tr> <td>R-----HI</td> <td>---GGDTVNA</td> </tr> </tbody> </table>	X1	Y2	EGEK---DG	GD--PEGRMMS	R-----YK	PQ--GPDRVNA	RGGTGGHDK	--ADGGDTFDS	R-----HI	---GGDTVNA	<table> <thead> <tr> <th>Y1</th> <th>X2</th> </tr> </thead> <tbody> <tr> <td>EGEK-DG--</td> <td>GDPEGRMMS</td> </tr> <tr> <td>--R--YK--</td> <td>PQGPDRVNA</td> </tr> <tr> <td>RGGTGGHDK</td> <td>ADGGDTFDS</td> </tr> <tr> <td>---R-----</td> <td>HIGGDTVNA</td> </tr> </tbody> </table>	Y1	X2	EGEK-DG--	GDPEGRMMS	--R--YK--	PQGPDRVNA	RGGTGGHDK	ADGGDTFDS	---R-----	HIGGDTVNA
X1	Y2																				
EGEK---DG	GD--PEGRMMS																				
R-----YK	PQ--GPDRVNA																				
RGGTGGHDK	--ADGGDTFDS																				
R-----HI	---GGDTVNA																				
Y1	X2																				
EGEK-DG--	GDPEGRMMS																				
--R--YK--	PQGPDRVNA																				
RGGTGGHDK	ADGGDTFDS																				
---R-----	HIGGDTVNA																				

Figure 5.2: Single point crossover operator: The first parent alignment (X) is cut straight at a randomly chosen position, and two blocks are created (X1|X2). The second parent alignment (Y) is tailored so that the right piece can be joined to the left piece of the first parent (Y1|Y2). Then, the blocks of (Y1|Y2) are filled with indels in order to match the size and keep the order of the letters in the sequences. In the end, the blocks of two parents are crossed (X1|Y2 and Y1|Y2)

5.5 Experimental Analysis

In this section, we describe the experimental analysis of PLS and IPLS with different parameters. Moreover, we compare them against our approach based on NSGA-II. At last, the alignments produced by the IPLS approach are compared with those produced by single-objective methods such as T-Coffee and Clustal Omega.

5.5.1 Performance Analysis of PLS and IPLS

In the following, we analyze the effectiveness of various parameters such as starting alignments, neighborhood, and perturbation, on the overall performance of PLS and IPLS. These experiments also allow us to understand the functional relationships between the performance of the algorithms and instance features, such as the number of sequences and their sizes.

The implementations were coded in C and compiled with GCC version 4.6.1 with the `-O3` compiler option. Experiments were performed on a cluster with 16 nodes, each one with 4-core Intel Core i7 CPU and 2 GB RAM, operating system Ubuntu 11.10. Except for the compiler option, no other code optimization techniques were used in the experiments.

We used all the 38 instances of RV11 and 41 instances of RV20 of benchmark BALiBASE 3.0. In our experiments, we considered PAM250 as the substitution matrix. For the initial alignments, three possibilities were examined: i) `Rand`, is a random feasible alignment with the least possible number of indels (see Section 5.4.1, page 108); ii) `T-Coffee`, is a consistency-based method that outperforms the others existing programs in terms of accuracy; iii) `Clust`, is a progressive method that outperforms T-Coffee in sequences with large N/C terminal extensions (Pais et al., 2014). We considered the default parameters for the last two programs.

We analyzed the effectiveness of neighborhood size by considering different values for k in the k -block neighborhood. For a given instance, let `Min` denote the length of the smallest sequence. We considered $k \in \{\text{Min}, \text{Min}/2, \text{Min}/4, \text{Min}/8, \text{Min}/16\}$. The PLS terminates once it is not possible to find non-dominated neighboring alignments or the time limit of 5 minutes of CPU-time is reached.

We evaluated the quality of each approximate set by computing its hypervolume indicator value (see Section 3.4.1, page 65), where the reference point for each instance is the minimum substitution score minus one and the maximum number of indels plus one, that was obtained from the runs of all local search variants. In order to obtain the reference

hypervolume indicator value, we merged all approximate sets collected in the experimental analysis, removed the dominated scores and computed the hypervolume of the remaining set. This value is used as a reference value to evaluate the relative performance of each approach. Then, for each approximate set, we computed the relative hypervolume indicator value (see Section 3.4.1, page 66).

Tables 5.2 and 5.3 report the results obtained for PLS on the benchmark sets RV11 and RV20, respectively. The results are averaged over 30 runs, for each of the five k values and the three starting alignments. Column `id` corresponds to the instance id from the two benchmark sets (`BB1100*.tfa` and `BB200*.tfa`, where $*$ denotes the id); column `m` gives the number of sequences; columns `Min` and `Max` correspond to the length of the smallest and the largest sequence, respectively. The instances are shown in the tables according to the number of sequences and the length of the smallest sequence. The values in bold correspond to the best result obtained for each instance. The last column `Init` shows the relative hypervolume indicator obtained from the non-dominated score of the two starting alignments: `Clustal Omega` and `T-Coffee`.

The results indicate that PLS has better performance when starting with alignments `Clust` and `T-Coffee`, instead of a feasible random alignment. Also, the best value for k strongly depends on the number of sequences and, to a smaller extent, to their sizes. As a rule, for the given cut-off time, large (low) k values achieve better performance on problems with a smaller (larger) number of sequences. Moreover, column `Init` indicates that PLS actively improves upon the two starting alignments.

Tables 5.4 and 5.5 report results obtained for IPLS on the same benchmark sets, averaged over 30 runs, for k -block size equals to 2; we recall that `Clust` and `T-Coffee` are used as starting alignments. To produce the perturbed alignments, we set $N_{in} = 5$, if the maximum length of the unaligned sequences is less than 100; otherwise, N_{in} is set to $1/20$ of the maximum length of the sequences. The results reported in the tables do not include the CPU-time taken to compute the starting alignments. IPLS always terminates once the time limit of 5 minutes of CPU-time is reached. Column `PLSmax` gives the best value of PLS from Tables 5.2 and 5.3. The best results for each sequence set are shown in bold. In most of the instances, IPLS improves over PLS, although the best performance may depend on the number of sequences and their sizes: a small (large) number of perturbations gives better performance on larger (smaller) sequences and larger (smaller) number of sequences.

Table 5.2: Average relative hypervolume indicator value (I_{RHV}) for PLS for several k -block neighborhood sizes and starting solutions (Clustal Omega, T-Coffee and Rand) for each instance of the dataset RV11. The results are averaged over 30 runs. See text for more details

id	m	$k = \text{Min}$			$k = \text{Min}/2$			$k = \text{Min}/4$			$k = \text{Min}/8$			$k = \text{Min}/16$			
		Min	Max	Rand	Clust	Tcoff	Rand	Clust	Tcoff	Rand	Clust	Tcoff	Rand	Clust	Tcoff	Init	
22	4	63	205	0.86	0.77	0.90	0.81	0.67	0.90	0.86	0.83	0.89	0.71	0.72	0.45	0.55	0.14
25	4	64	103	0.53	0.84	0.84	0.54	0.76	0.86	0.51	0.86	0.74	0.86	0.48	0.55	0.49	0.28
29	4	81	138	0.69	0.81	0.85	0.67	0.85	0.88	0.69	0.65	0.87	0.85	0.65	0.72	0.81	0.35
1	4	83	91	0.17	0.27	0.35	0.17	0.25	0.35	0.16	0.13	0.34	0.34	0.10	0.28	0.33	0.20
9	4	97	337	0.61	0.87	0.78	0.61	0.71	0.67	0.60	0.59	0.44	0.22	0.53	0.30	0.10	0.19
21	4	102	139	0.51	0.91	0.86	0.54	0.86	0.90	0.53	0.54	0.81	0.76	0.52	0.61	0.72	0.18
8	4	104	540	0.79	0.89	0.84	0.79	0.89	0.82	0.79	0.80	0.86	0.81	0.74	0.33	0.75	0.03
17	4	247	264	0.37	0.90	0.88	0.33	0.90	0.91	0.33	0.31	0.88	0.84	0.27	0.74	0.78	0.36
15	4	297	327	0.46	0.92	0.87	0.44	0.93	0.89	0.53	0.49	0.93	0.94	0.45	0.90	0.92	0.38
12	4	320	397	0.38	0.91	0.87	0.36	0.92	0.90	0.44	0.44	0.90	0.87	0.36	0.84	0.82	0.33
24	4	372	465	0.50	0.89	0.83	0.50	0.90	0.90	0.50	0.50	0.76	0.73	0.51	0.66	0.68	0.17
4	4	390	456	0.35	0.91	0.85	0.36	0.91	0.88	0.36	0.35	0.89	0.85	0.35	0.76	0.77	0.23
3	4	414	516	0.43	0.92	0.88	0.42	0.90	0.90	0.41	0.42	0.90	0.90	0.42	0.84	0.88	0.38
10	4	490	492	0.03	0.95	0.85	0.03	0.91	0.84	0.03	0.01	0.82	0.80	0.01	0.67	0.73	0.25
13	5	51	101	0.69	0.80	0.83	0.69	0.81	0.87	0.69	0.66	0.83	0.80	0.63	0.58	0.67	0.06
35	5	71	138	0.72	0.84	0.81	0.71	0.85	0.79	0.74	0.85	0.80	0.79	0.67	0.62	0.77	0.14
11	5	160	242	0.54	0.94	0.87	0.54	0.94	0.90	0.52	0.50	0.90	0.85	0.53	0.69	0.76	0.10
37	5	335	1192	0.38	0.68	0.67	0.46	0.73	0.74	0.54	0.62	0.81	0.88	0.70	0.81	0.93	0.24
14	6	502	634	0.39	0.90	0.85	0.36	0.93	0.90	0.47	0.37	0.96	0.96	0.50	0.95	0.95	0.52
26	7	76	906	0.61	0.75	0.77	0.76	0.79	0.85	0.88	0.89	0.91	0.92	0.96	0.66	0.82	0.13
27	7	175	432	0.48	0.82	0.72	0.57	0.88	0.78	0.65	0.71	0.95	0.84	0.68	0.91	0.79	0.24
23	7	231	407	0.54	0.81	0.76	0.61	0.86	0.81	0.66	0.62	0.93	0.89	0.72	0.91	0.83	0.37
2	8	52	193	0.85	0.87	0.88	0.84	0.86	0.87	0.85	0.84	0.86	0.83	0.82	0.67	0.78	0.26
6	8	186	283	0.31	0.80	0.73	0.31	0.86	0.77	0.32	0.32	0.88	0.88	0.33	0.80	0.86	0.28
32	8	226	403	0.45	0.80	0.76	0.49	0.85	0.82	0.55	0.57	0.92	0.88	0.60	0.93	0.90	0.40
38	8	261	614	0.23	0.79	0.76	0.28	0.82	0.80	0.35	0.47	0.90	0.92	0.54	0.91	0.94	0.37
36	8	298	436	0.38	0.78	0.68	0.43	0.83	0.73	0.44	0.49	0.92	0.85	0.48	0.94	0.87	0.41
16	8	316	729	0.27	0.71	0.77	0.32	0.76	0.82	0.38	0.50	0.83	0.89	0.58	0.88	0.95	0.37
34	8	401	729	0.27	0.69	0.79	0.30	0.72	0.83	0.36	0.46	0.85	0.89	0.54	0.81	0.94	0.32
20	9	201	237	0.29	0.87	0.76	0.30	0.90	0.83	0.30	0.92	0.87	0.88	0.27	0.86	0.89	0.40
7	9	385	457	0.37	0.79	0.68	0.39	0.84	0.77	0.38	0.41	0.92	0.89	0.41	0.93	0.89	0.37
28	10	93	211	0.53	0.90	0.85	0.56	0.91	0.90	0.56	0.56	0.89	0.86	0.51	0.77	0.82	0.40
19	10	299	396	0.35	0.74	0.74	0.38	0.80	0.76	0.40	0.42	0.91	0.87	0.44	0.91	0.90	0.43
33	11	85	239	0.57	0.78	0.82	0.62	0.82	0.83	0.64	0.67	0.80	0.92	0.66	0.75	0.88	0.26
31	11	300	611	0.22	0.77	0.73	0.26	0.80	0.75	0.31	0.40	0.88	0.84	0.67	0.96	0.84	0.21
30	14	236	392	0.23	0.73	0.76	0.28	0.77	0.76	0.31	0.37	0.80	0.85	0.40	0.90	0.89	0.51
5	14	329	465	0.20	0.70	0.71	0.21	0.73	0.73	0.26	0.30	0.82	0.79	0.31	0.85	0.85	0.53
18	14	418	750	0.09	0.71	0.87	0.12	0.73	0.88	0.15	0.19	0.79	0.88	0.24	0.82	0.90	0.45

Table 5.4: Average relative hypervolume indicator value (I_{RHV}) for IPLS with several perturbation iterations (P) for each instance of dataset RV11. The results are averaged over 30 runs. See text for more details

id	m	Min	Max	PLS ^{max}	P = 1	P = 2	P = 4	P = 8	P = 16
22	4	63	205	0.90	0.91	0.87	0.89	0.88	0.97
25	4	64	103	0.86	0.86	0.89	0.91	0.89	0.87
29	4	81	138	0.88	0.91	0.91	0.94	0.92	0.98
1	4	83	91	0.35	0.36	0.38	0.40	0.42	0.44
9	4	97	337	0.87	0.87	0.81	0.87	0.83	0.86
21	4	102	139	0.91	0.95	0.89	0.92	0.88	0.93
8	4	104	540	0.89	0.87	0.91	0.97	0.88	0.90
17	4	247	264	0.91	0.92	0.93	0.95	0.96	0.97
15	4	297	327	0.94	0.90	0.92	0.96	0.97	0.97
12	4	320	397	0.92	0.92	0.91	0.96	0.97	0.99
24	4	372	465	0.90	0.87	0.85	0.86	0.83	0.80
4	4	390	456	0.91	0.93	0.87	0.88	0.88	0.84
3	4	414	516	0.92	0.92	0.89	0.89	0.88	0.88
10	4	490	492	0.95	0.97	0.95	0.96	0.90	0.94
13	5	51	101	0.87	0.91	0.92	0.93	0.93	0.95
35	5	71	138	0.85	0.86	0.89	0.88	0.93	0.99
11	5	160	242	0.94	0.94	0.89	0.91	0.88	0.91
37	5	335	1192	0.93	0.92	0.94	0.91	0.99	0.98
14	6	502	634	0.96	0.95	0.97	0.95	0.94	0.91
26	7	76	906	0.96	0.99	0.96	0.95	0.66	0.60
27	7	175	432	0.95	0.91	0.87	0.85	0.85	0.69
23	7	231	407	0.93	0.89	0.81	0.82	0.82	0.79
2	8	52	193	0.88	0.92	0.84	0.82	0.84	0.78
6	8	186	283	0.88	0.88	0.77	0.75	0.75	0.66
32	8	226	403	0.93	0.88	0.85	0.83	0.81	0.74
38	8	261	614	0.94	0.86	0.88	0.89	0.85	0.81
36	8	298	436	0.94	0.94	0.94	0.94	0.96	0.97
16	8	316	729	0.95	0.94	0.93	0.96	0.95	0.99
34	8	401	729	0.94	0.85	0.83	0.81	0.80	0.77
20	9	201	237	0.92	0.96	0.97	0.96	0.90	0.83
7	9	385	457	0.93	0.92	0.92	0.97	0.93	0.88
28	10	93	211	0.91	0.95	0.95	0.90	0.87	0.86
19	10	299	396	0.91	0.85	0.85	0.83	0.83	0.81
33	11	85	239	0.92	0.92	0.91	0.90	0.87	0.84
31	11	300	611	0.96	0.95	0.88	0.89	0.84	0.82
30	14	236	392	0.90	0.89	0.86	0.81	0.80	0.78
5	14	329	465	0.85	0.87	0.83	0.80	0.75	0.75
18	14	418	750	0.90	0.94	0.92	0.90	0.88	0.87

Table 5.5: Average relative hypervolume indicator value (I_{RHV}) for IPLS with several perturbation numbers (P) for each instance of the dataset RV20. The results are averaged over 30 runs. See text for more details

id	m	Min	Max	PLS ^{max}	P = 1	P = 2	P = 4	P = 8	P = 16
20	16	74	697	0.95	0.95	0.95	0.88	0.83	0.73
1	16	247	527	0.90	0.90	0.90	0.87	0.88	0.85
2	20	52	1520	0.82	0.93	0.94	0.94	0.89	0.84
11	21	95	631	0.90	0.97	0.95	0.91	0.86	0.85
7	23	381	457	0.93	0.95	0.96	0.95	0.89	0.83
19	24	401	729	0.96	0.91	0.94	0.95	0.95	0.87
16	27	106	458	0.89	0.91	0.90	0.80	0.78	0.72
12	27	210	634	0.89	0.89	0.82	0.80	0.59	0.52
13	28	447	747	0.94	0.69	0.77	0.86	0.83	0.87
29	29	64	167	0.83	0.89	0.84	0.82	0.81	0.81
9	29	254	415	0.94	0.89	0.96	0.94	0.82	0.76
27	29	279	1052	0.87	0.73	0.81	0.87	0.90	0.92
10	29	577	1233	0.92	0.64	0.73	0.80	0.90	0.94
24	30	74	739	0.86	0.80	0.85	0.90	0.91	0.88
23	31	202	244	0.92	0.98	0.95	0.91	0.87	0.73
26	32	271	1016	0.86	0.73	0.85	0.90	0.93	0.82
35	35	226	982	0.83	0.80	0.78	0.82	0.88	0.90
15	37	54	274	0.94	0.82	0.91	0.94	0.86	0.80
38	42	79	199	0.94	0.94	0.94	0.84	0.73	0.69
5	42	343	474	0.97	0.83	0.85	0.87	0.91	0.93
17	45	509	713	0.97	0.71	0.74	0.81	0.87	0.94
30	47	76	155	0.84	0.84	0.87	0.83	0.80	0.83
33	48	81	155	0.89	0.95	0.89	0.81	0.67	0.69
41	48	293	1520	0.64	0.88	0.84	0.82	0.83	0.86
6	51	224	293	0.95	0.85	0.91	0.95	0.93	0.86
18	53	296	381	0.96	0.70	0.87	0.94	0.83	0.67
21	53	418	838	0.83	0.89	0.91	0.93	0.94	0.91
28	54	241	610	0.97	0.96	0.97	0.96	0.94	0.93
4	55	401	734	0.93	0.81	0.80	0.84	0.89	0.94
8	56	98	1520	0.72	0.72	0.78	0.81	0.85	0.87
22	58	320	381	0.89	0.62	0.68	0.79	0.91	0.92
34	59	234	548	0.85	0.75	0.75	0.82	0.88	0.86
31	60	175	432	0.92	0.87	0.87	0.83	0.82	0.88
32	61	93	149	0.97	0.97	0.98	0.88	0.73	0.67
37	65	160	810	0.84	0.80	0.84	0.81	0.80	0.91
14	65	333	729	0.66	0.84	0.84	0.83	0.86	0.90
3	74	409	800	0.88	0.73	0.90	0.81	0.76	0.95
25	81	372	518	0.87	0.82	0.86	0.85	0.81	0.96
40	87	273	570	0.87	0.83	0.92	0.91	0.88	0.94
36	91	82	335	0.81	0.82	0.83	0.77	0.78	0.92
39	91	298	458	0.85	0.73	0.76	0.79	0.81	0.85

5.5.2 Comparison of IPLS with NSGA-II

In this section, we compare our IPLS approach with NSGA-II as described in Section 5.4.3. To create the initial solutions for both algorithms, we consider the pre-computed alignments introduced in Table 5.1. We use the default parameters suggested for these programs. In the IPLS approach, the initial set of alignments Φ_0 were filled up directly with the obtained alignments from Table 5.1. In the NSGA-II, we used the approach described in Section 5.4.3. In addition, we set the population size to 100, and used the mutation and crossover operators as explained in Section 5.4.3, page 113. We set the probabilities of 0.8 and 0.2 for the single-point crossover and gap shifting mutation operators as it is proposed by [Ortuño et al. \(2013\)](#), respectively.

We selected a subset of the benchmark BALiBASE 3.0, in particular, we chose the first ten datasets from the families RV11, RV12, RV20, RV30, RV40 and RV50, which sums up to 60 problem instances. The stopping condition was set to 50,000 function evaluations for both methods (NSGA-II and IPLS). PAM250 is used as the substitution matrix.

We used relative hypervolume indicator and the empirical attainment functions (EAF). The hypervolume indicator was applied in the same way as described in the previous section. The EAF was mainly used to visualize the differences of performance in the objective space.

The best results for each problem instances are shown with bold faces in Table 5.6. From these tables, a first observation is that IPLS obtains the best average values for a more significant number of instances. In almost all the sample instances of RV20, RV30, RV40, and RV50 families, IPLS has larger indicator values than NSGA-II. In addition, IPLS in 80% of the instances in RV11 and 60% of the instances in RV12 has better indicator values. IPLS presents better results when the sequences instances are more similar. Note that RV11 contains datasets with less than 20% residue identity between groups and RV12 contains datasets with residue identity between 20% and 40%. The remaining dataset groups comprise family sequences with more than 40% similarity.

We used the EAF tools² to compute bi-dimensional EAFs from the collection of approximation sets and to plot the attainment surfaces as well as the differences in terms of EAFs produced by pairs of heuristics ([López-Ibañez et al., 2010](#)). Figure 5.3 and 5.4 show the EAF difference plot for instances BB11008 and BB20003, respectively. The differences in favor of NSGA II are shown in the left plots whereas those in favor of IPLS are shown in the right plots. The EAF difference plot, for instance, BB20003 shows the superiority of IPLS over

²EAFtools is an R script, available at <http://lopez-ibanez.eu/eaftools>.

NSGA-II in the objective space. However, for instance, BB11008, NSGA-II performs better on the minimization of indels. These two findings are consistent with those obtained in the remaining instances. Appendix A reports the EAF plots obtained in the 60 instances.

5.5.3 Comparison of IPLS with Clustal Omega and T-Coffee

In order to gain insight into the absolute quality of the alignments produced by IPLS, we compared the alignments produced by IPLS with the reference alignments in the BALiBASE benchmark (Thompson et al., 1999). The BALiBASE dataset defines a well-known benchmark to standardize the comparison of sequence alignment results. It consists of a group of protein sequences that are properly prepared to be aligned by MSA algorithms (see Section 2.4.4, page 40). We rely on the correctly aligned residue pairs (SP) and correctly aligned columns (TC) measures (see Section 2.4.3, page 37). We used seven different datasets from BALiBASE with specific features chosen from this benchmark (see Table 5.7). Columns `Reference` and `id` correspond to the reference and id number of the dataset in the BALiBASE benchmark. Columns `m`, `Min` and `Max` correspond to the number of sequences, length of the smallest and largest sequence, respectively. `Len` corresponds to the size of reference alignment in the BALiBASE benchmark.

We ran IPLS 30 times on these datasets, and for each collection of runs, we chose the alignment with the least number of indels. We computed the SP and TC ratios by using this alignment and the reference alignment available for each chosen dataset in BALiBASE. For the calculation of the SP ratio, we used the substitution matrix PAM 250. Moreover, for comparison purpose, we performed the same procedure for the alignments produced by Clustal Omega and T-Coffee.

Tables 5.8 and 5.9 show the results obtained with SP and TC ratios, respectively. The boldface values represent the best ratio. From Table 5.8, it can be observed that IPLS obtained the best value for all the tested datasets. In Table 5.9, the best TC ratio is either from T-Coffee or Clustal Omega. Note that a null TC value was obtained by the alignment of Clustal Omega in RV12 id 20 and RV20 id 20, and by T-Coffee in set RV20 and id 1. In general, IPLS has better results from both Clustal Omega and T-Coffee in terms of SP ration. Clustal Omega and T-Coffee have better outcomes in TC ratio. However, there are cases that either Clustal Omega or T-Coffee have null values, while the IPLS has positive results.

Table 5.6: Average relative hypervolume indicator value (I_{RHV}) for IPLS and NSGA-II for the first 10 instances of the datasets RV11, RV12, RV20, RV30, RV40 and RV50. The results are averaged over 30 runs for each algorithm. See text for more details

RV11			RV12		
id	NSGA-II	IPLS	id	NSGA-II	IPLS
BB11001	0.91 ± 0.03	0.89 ± 0.01	BB12001	0.95 ± 0.01	0.95 ± 0.01
BB11002	0.95 ± 0.01	0.96 ± 0.01	BB12002	0.90 ± 0.04	0.89 ± 0.04
BB11003	0.93 ± 0.02	0.94 ± 0.02	BB12003	0.88 ± 0.04	0.85 ± 0.03
BB11004	0.95 ± 0.01	0.96 ± 0.01	BB12004	0.89 ± 0.04	0.91 ± 0.04
BB11005	0.96 ± 0.01	0.97 ± 0.01	BB12005	0.92 ± 0.02	0.93 ± 0.02
BB11006	0.93 ± 0.01	0.95 ± 0.02	BB12006	0.93 ± 0.02	0.91 ± 0.03
BB11007	0.93 ± 0.01	0.94 ± 0.02	BB12007	0.94 ± 0.02	0.96 ± 0.01
BB11008	0.93 ± 0.01	0.92 ± 0.02	BB12008	0.89 ± 0.02	0.91 ± 0.02
BB11009	0.95 ± 0.00	0.96 ± 0.01	BB12009	0.92 ± 0.02	0.94 ± 0.02
BB11010	0.94 ± 0.01	0.94 ± 0.01	BB12010	0.93 ± 0.02	0.93 ± 0.03

RV20			RV30		
id	NSGA-II	IPLS	id	NSGA-II	IPLS
BB20001	0.86 ± 0.03	0.90 ± 0.02	BB30001	0.85 ± 0.04	0.90 ± 0.03
BB20002	0.97 ± 0.03	0.98 ± 0.00	BB30002	0.93 ± 0.02	0.93 ± 0.02
BB20003	0.85 ± 0.03	0.92 ± 0.02	BB30003	0.84 ± 0.03	0.89 ± 0.04
BB20004	0.91 ± 0.03	0.93 ± 0.01	BB30004	0.92 ± 0.02	0.94 ± 0.02
BB20005	0.94 ± 0.02	0.96 ± 0.02	BB30005	0.89 ± 0.03	0.90 ± 0.04
BB20006	0.91 ± 0.02	0.93 ± 0.02	BB30006	0.91 ± 0.03	0.91 ± 0.02
BB20007	0.94 ± 0.01	0.95 ± 0.02	BB30007	0.93 ± 0.01	0.94 ± 0.01
BB20008	0.97 ± 0.00	0.99 ± 0.00	BB30008	0.94 ± 0.02	0.96 ± 0.02
BB20009	0.91 ± 0.03	0.92 ± 0.02	BB30009	0.93 ± 0.02	0.93 ± 0.02
BB20010	0.88 ± 0.04	0.93 ± 0.04	BB30010	0.91 ± 0.02	0.94 ± 0.02

RV40			RV50		
id	NSGA-II	IPLS	id	NSGA-II	IPLS
BB40001	0.93 ± 0.02	0.96 ± 0.03	BB50001	0.95 ± 0.01	0.96 ± 0.01
BB40002	0.95 ± 0.01	0.96 ± 0.01	BB50002	0.98 ± 0.00	0.99 ± 0.00
BB40003	0.92 ± 0.02	0.95 ± 0.02	BB50003	0.96 ± 0.01	0.98 ± 0.01
BB40004	0.91 ± 0.03	0.92 ± 0.03	BB50004	0.95 ± 0.01	0.96 ± 0.01
BB40005	0.91 ± 0.03	0.94 ± 0.03	BB50005	0.93 ± 0.01	0.95 ± 0.01
BB40006	0.92 ± 0.02	0.95 ± 0.02	BB50006	0.95 ± 0.01	0.96 ± 0.01
BB40007	0.96 ± 0.01	0.97 ± 0.01	BB50007	0.91 ± 0.02	0.93 ± 0.03
BB40008	0.92 ± 0.01	0.95 ± 0.01	BB50008	0.91 ± 0.02	0.92 ± 0.03
BB40009	0.94 ± 0.01	0.95 ± 0.01	BB50009	0.93 ± 0.01	0.95 ± 0.01
BB40010	0.83 ± 0.05	0.81 ± 0.06	BB50010	0.95 ± 0.00	0.98 ± 0.00

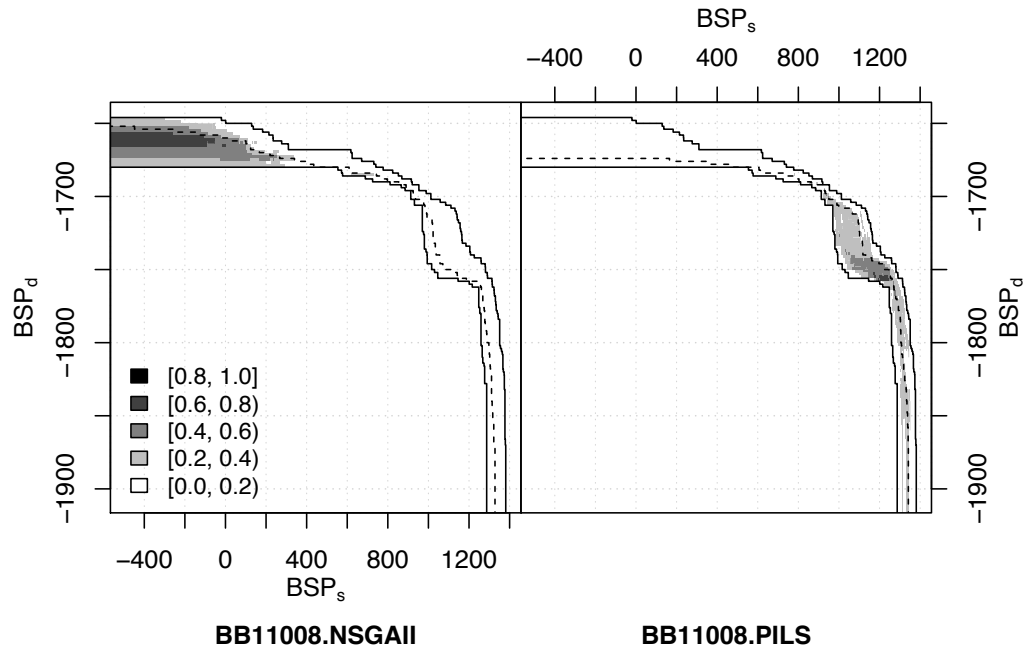


Figure 5.3: EAF difference plot for instance BB11008 from family group RV11.

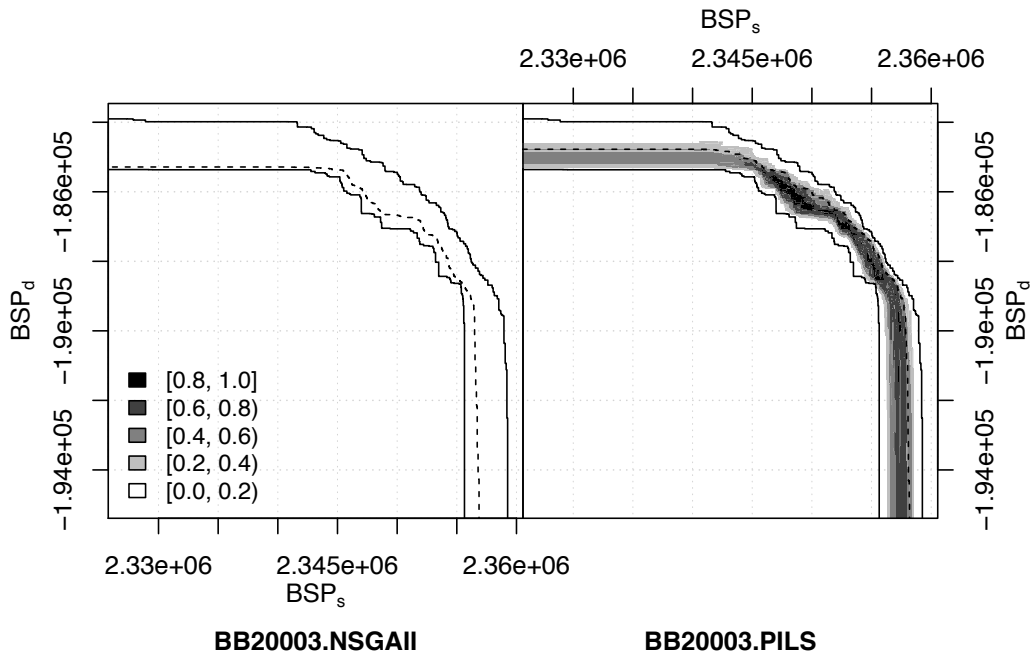


Figure 5.4: EAF difference plot for instance BB20003 from family group RV12.

Table 5.7: The selective datasets from difference reference of the benchmark.

Reference	id	Identity	m	Min	Max	Len
RV11	1	<20%	4	83	91	96
RV11	4	<20%	4	390	456	603
RV12	20	<20% & >40%	4	118	129	141
RV12	42	<20% & >40%	4	448	561	611
RV20	20	>40%	16	74	697	615
RV20	1	>40%	16	247	527	780
RV30	17	–	15	231	370	416
RV40	10	–	9	67	214	275
RV40	14	–	9	298	609	712
RV50	4	–	9	386	505	547

Table 5.8: The results of SP score in T-Coffee, Clustal Omega and IPLS on the selected test cases.

Reference	id	Clustal Omega	T-Coffee	IPLS
RV11	1	0.956	0.965	0.985
RV11	4	0.033	0.706	0.788
RV12	20	0.331	0.973	0.981
RV12	42	0.678	0.789	0.85
RV20	20	0.535	0.934	0.946
RV20	1	0.939	0.596	0.951
RV30	17	0.765	0.787	0.811
RV40	10	0.875	0.872	0.878
RV40	14	0.878	0.890	0.894
RV50	4	0.973	0.983	0.988

Table 5.9: The results of TC score in T-Coffee, Clustal Omega and IPLS on the selected test cases.

Reference	id	Clustal Omega	T-Coffee	IPLS
RV11	1	0.912	0.930	0.885
RV11	4	0.408	0.554	0.378
RV12	20	0.000	0.957	0.851
RV12	42	0.548	0.662	0.434
RV20	20	0.000	0.775	0.654
RV20	1	0.775	0.000	0.459
RV30	17	0.552	0.581	0.535
RV40	10	0.639	0.590	0.330
RV40	14	0.671	0.658	0.463
RV50	4	0.919	0.940	0.892

5.6 Summary and Discussion

In this chapter, we formulated the multiobjective multiple sequence alignment problem and proposed a Pareto Local Search (PLS) algorithm for this problem according to a given definition of neighborhood. Furthermore, we extended this approach to iterated local search with a particular definition of perturbation (IPLS).

We analyzed the effect of various parameters such as starting alignments, neighborhood, and perturbation on the overall performance of PLS and IPLS. The results indicate that PLS improves upon the starting alignments and IPLS has better performance than PLS. Moreover, we compared our IPLS approach with our adaptation of NSGA-II. From the experimental analysis, we have observed that IPLS obtains the best performance in terms of relative hypervolume in a large number of instances. At last, we compared the IPLS with two single-objective methods: Clustal Omega and T-Coffee. From the chosen set of alignments from the reference benchmark, we observed that the alignment obtained from IPLS with the least number of indels has a better ratio in terms of SP and it has competitive results in terms of TC score.

For the future work, we would like to consider other starting alignments that are not just based on the similarity but also take into account other biological criteria such as the secondary structure of the alignments. Further, we may test different perturbation methods. One of the possibilities is to use the mutation methods that is proposed in genetic algorithms such as the one used in MOSAStrE (Ortuño et al., 2013).

In the next chapter, the other main contribution of this thesis is introduced, in particular, the application of the multiobjective formulation of sequence alignment in the construction of phylogenetic trees.

Chapter 6

An Application to the Construction of Phylogenetic Trees

6.1 Introduction

Phylogenetic trees are diagrams that represent the evolutionary relationships among different groups of organisms, or among a family of related nucleic acid or protein sequences, e.g., how might have this family been obtained during evolution (Sleator, 2011). The construction of phylogenetic trees requires multiple sequence alignment methods to align genetic data.

Different alignments may lead to different phylogenetic trees and, consequently, to different possible relationships between the organisms under study. In fact, Wong et al. (2008) recently have recently shown that different alignment programs often lead to different tree topologies. Our working hypothesis is that more information can be extracted from Pareto optimal alignments, leading to different tree topologies. This would help the biologist to describe the inferred phylogenies better.

In this chapter, we describe a method for constructing phylogenetic trees from the non-dominated score set obtained from all pairwise combinations of sequences. In the context of the multiobjective framework, it is important to understand whether the non-dominated score set can provide further information than that produced by known methods. The main

contribution of this chapter is proposing a new method for the construction of phylogenetic trees based on the multiobjective sequence alignment. Moreover, we illustrate the application of this method on known sequence data to show the effectiveness of this approach in practice. The resulted publications are as follows: the publication of a journal article (Abbasi et al., 2013a) and two posters presented at international conferences (Abbasi et al., 2011, 2012) is the proposal of a novel approach to constructing the phylogenetic trees. Furthermore, we discuss its application for deriving further information about the reliability of the tree branches.

This chapter is organized as follows. We describe the required definitions, notations and the main steps to construct a phylogenetic tree in Section 6.2. Section 6.3 presents the main methods that can be applied to create a phylogenetic tree from given input sequences. Section 6.4 introduces our biobjective method for constructing phylogenetic trees. In Section 6.5, two data sets are used for illustration purpose, and the resulting trees are compared with those obtained with a well-known technique. At last, we present the main conclusions and discussion in Section 6.6.

6.2 Introduction to Phylogenetic Trees

Phylogeny refers to the evolutionary history of species. *Phylogenetics* is the science that explains the evolutionary relationship between species (Randall T. Schuh, 2009). In the molecular phylogenetic analysis, the sequence of a common genome or protein can be used to assess the evolutionary relationship among species. The evolutionary relationship obtained from the phylogenetic analysis usually describes as *phylogenetic tree*. A phylogenetic tree or evolutionary tree is a diagrammatic representation of the evolutionary relationships among various species. It is a branching diagram composed of nodes and branches.

In phylogenetic trees, the terminal nodes (leaves) represent the operational taxonomic units (OTUs), which are the actual objects under analysis such as the species, populations, genome or protein sequences. The internal nodes represent hypothetical taxonomic units (HTUs) that show the last common ancestor to the nodes starting from this point. A branch or an edge describes the time estimate of the evolutionary relationships among the taxonomic units. One branch can connect only two nodes. Phylogenetic trees can be rooted or unrooted. A rooted tree has a node, the root, from which the rest of the tree diverges and corresponds to the last common ancestor. Figure 6.1 shows different forms of phylogenetic trees; (A) and (C) are rooted trees, while (B) is an unrooted tree. The tree in (C) is a dendrogram obtained

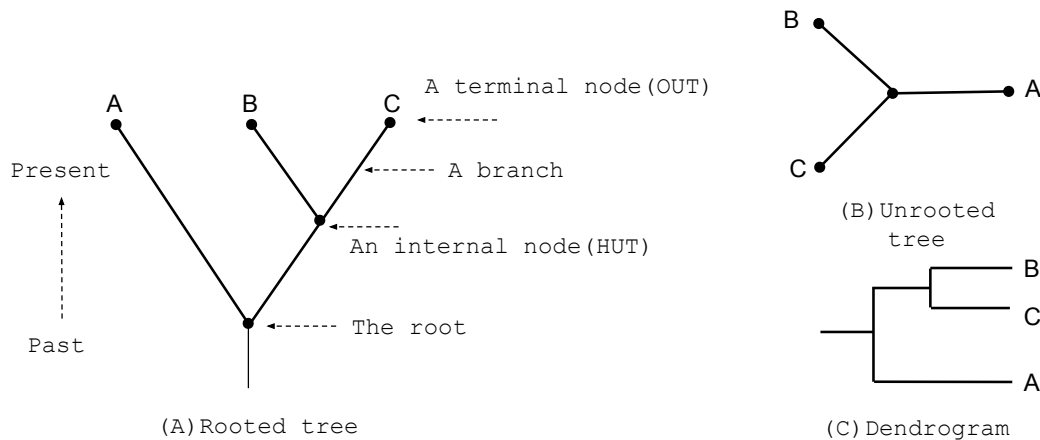


Figure 6.1: Different representations of a phylogenetic tree.

from hierarchical clustering methods.

Several methods exist to construct/reconstruct phylogenetic trees. However, they deal with certain assumptions such as *i*) the sequences are homologous, that is, the sequences share a common ancestry, and they have diverged through time as they evolved; and *ii*) each position inside the sequences evolved independently.

In general, construction of a phylogenetic tree consists of the following steps:

- i*) *Selection of a proper molecular marker (genomes/proteins)*. A molecular marker in phylogenetic analysis is the biological information that is used to infer the evolutionary relationships among taxa. Depending on the need, nucleic acid or protein sequences can be chosen as the appropriate marker.
- ii*) *Sequence alignment*. Alignment of sequences is the most critical step in constructing a reliable phylogenetic tree. Alignment identifies blocks of conserved residues.
- iii*) *Selection of a model of evolution*. An evolutionary model of sequence data is a model of nucleotide or amino acid substitution and resulting divergence of sequences. The simplest way to determine divergence is to count the number of substitutions. The simplest substitution model is the Bishop Friday model (Bishop and Friday, 1985) which assumes that all amino acids substitutions occur at an equal frequency and all substitutions occur at the same rate.
- iv*) *Construction of the phylogenetic tree*. Many methods have been described for reconstructing phylogenetic trees. The methods can be classified into two major types: *i*)

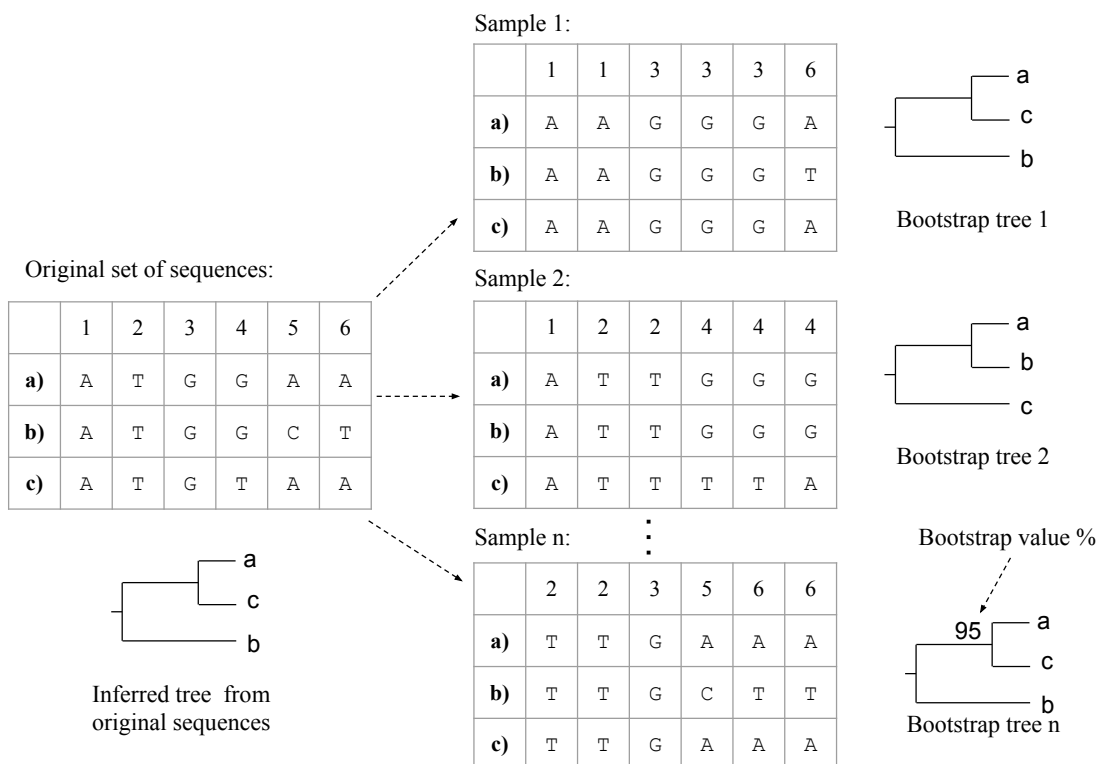


Figure 6.2: Illustration of bootstrapping method for a phylogenetic tree.

distance-based and *ii*) character-based. In Section 6.3, some of these methods are explained in more detail.

- v) *Assessment of the reliability of the tree.* Determining the reliability of a tree means deciding whether the topology of the tree is accurate or a better tree can be obtained. This question is answered by *bootstrapping* the tree. Bootstrapping, introduced in [Felsenstein \(1985\)](#), is a resampling analysis that involves generating a series of sequence alignments by sampling, with replacement, columns from the original sequence alignment. Each of these alignments is then used to rebuild the tree through the same analysis as the original sample to obtain many bootstrap trees. At last, the topology of these bootstrap trees is compared with that of the original to assess the reliability of the original phylogenetic tree. A bootstrapping may re-sample 500-1000 trees from the original sequences. If, for example, the same node recovered through 95 out of 100 iterations of taking out residues and resampling the tree, then the bootstrap value would be 0.95 which indicates that the node is well supported. A node that is well

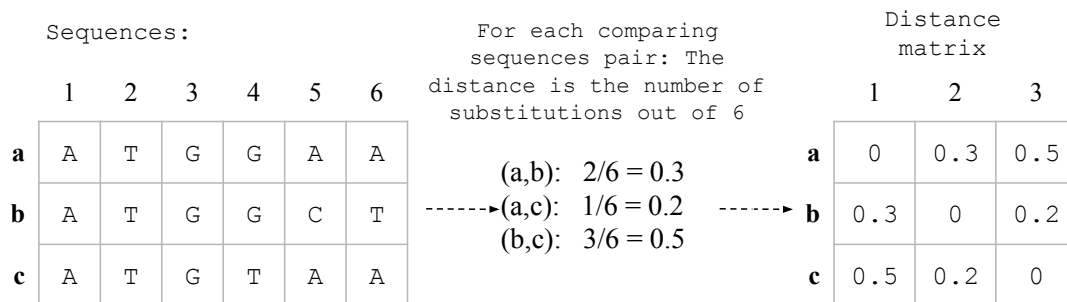


Figure 6.3: The distance matrix of three sequences with length 6. The distance is calculated as the number of substitutions between sequences divided by their lengths.

supported suggests that most of the residues support that node as removing them at random would not lead to a different reconstruction of that node. Figure 6.2 illustrates the principle of bootstrapping.

6.3 Construction of the Phylogenetic Tree

Phylogenetic tree construction methods can be categorized in two ways: distance and character-based methods (Sleator, 2011). The most common distance-based methods are UPGMA (Sneath et al., 1975) and Neighbor-Joining (Saitou and Nei, 1987). Both of these algorithms are based on a distance matrix that expresses “genetic distance” between the sequences. The alternative to these methods is the character based methods such as Maximum Parsimony (Fitch, 1971) or Maximum Likelihood (Felsenstein, 1981), which take a probabilistic approach. The following sections describe these methods in more details.

6.3.1 UPGMA

The UPGMA (Sneath et al., 1975) is one of the first distance-matrix methods that uses sequential clustering to build a rooted phylogenetic tree. First, all sequences are compared through pairwise alignment to compute a distance matrix. Figure 6.3 shows a simple computation of a distance matrix for 3 sequences. Using this matrix, the two sequences with minimum distance are identified and clustered as a single pair. Next, the distance between this joined pair and all other sequences is recalculated. Using these new distances, the process is repeated until all sequences have been included in the cluster. The result of UPGMA can be visualized as a rooted dendrogram. In the following, we illustrate this procedure with

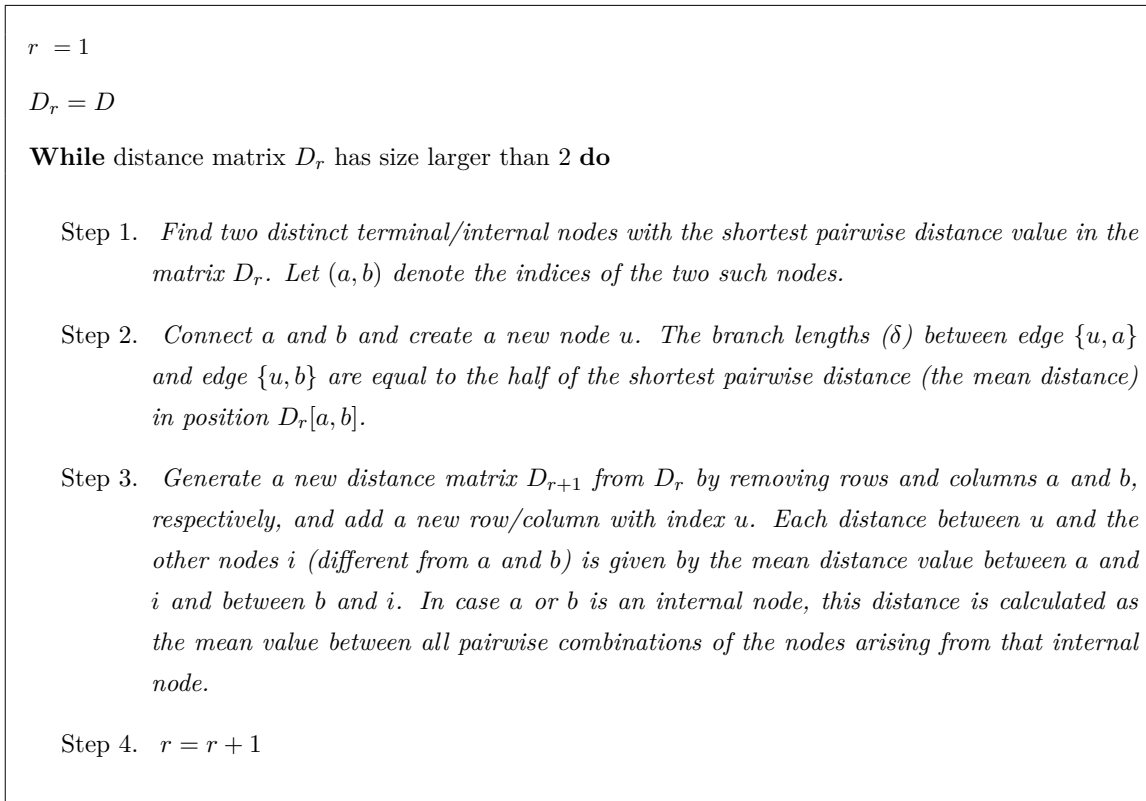


Figure 6.4: Steps of the UPGMA method

an example.

Consider n sequences and a distance matrix D of size $n \times n$ where $D[a, b]$ is the distance between the a -th sequence and the b -th sequence. The goal is to construct a rooted tree that matches the information from the distance matrix. The UPGMA algorithm starts from the two closest nodes and repeats the five steps until all the nodes are connected and branch lengths specified. Figure 6.4 shows the main steps of this method.

Example 6.1. Consider 5 sequences (a, b, c, d, e) with the distance matrix D . The algorithm works as follows:

	a	b	c	d	e
a	0	5	4	7	6
b	5	0	7	10	9
c	4	7	0	7	6
d	7	10	7	0	5
e	6	9	6	5	0

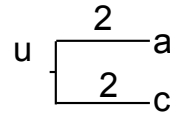
Copy matrix D to D_1 .

Iteration ($r \leftarrow 1$)

Step 1. The minimum value in the distance matrix D_1 corresponds to the pair (a, c) with value 4.

Step 2. The tree starts by connecting (a, c) through a new node u . The new node u will have the following branch lengths from a and c .

$$\begin{aligned} \delta[a, u] = \delta[c, u] &= D_1[a, c]/2 \\ &= 4/2 = 2 \end{aligned}$$



Step 3. Generate the matrix D_2 from D_1 . The two indices $(a$ and $c)$ are removed and replaced by the new index (u) . Each distance between u and the other nodes i (different from a and b) is given by the distance between a and i plus the distance between b and i , divided by two.

$$D_2[b, u] = (D[a, b] + D[c, b]) / 2 = (5 + 7) / 2 = 6$$

$$D_2[d, u] = (D[a, d] + D[c, d]) / 2 = (7 + 7) / 2 = 7$$

$$D_2[e, u] = (D[a, e] + D[c, e]) / 2 = (6 + 6) / 2 = 6$$

Therefore, the new distance matrix D_2 becomes:

	u	b	d	e
u	0	6	7	6
b	6	0	10	9
d	7	10	0	5
e	6	9	5	0

Step 4. $r \leftarrow 2$

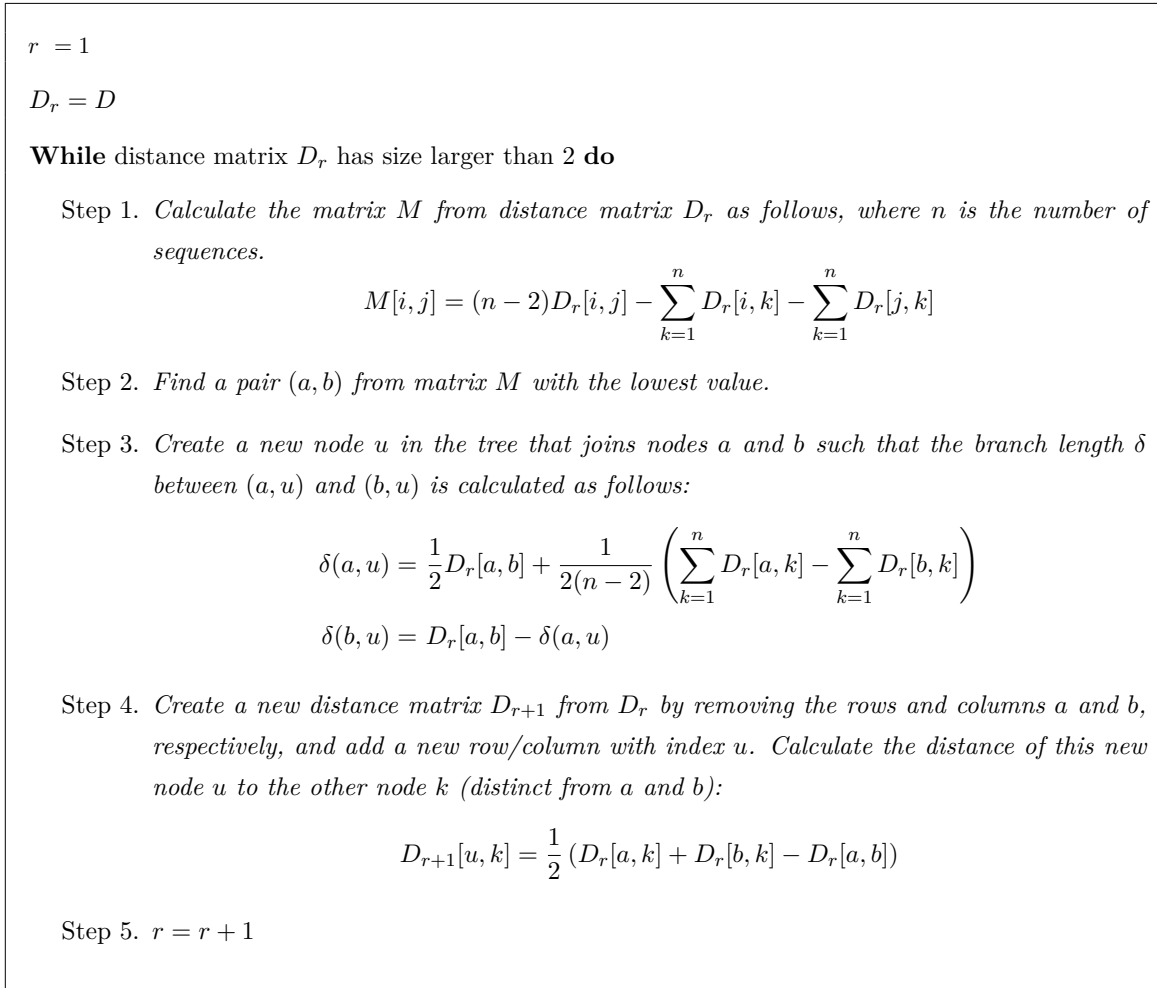


Figure 6.5: Steps of Neighbor Joining method

Repeat all the steps until the tree topology is fully resolved.

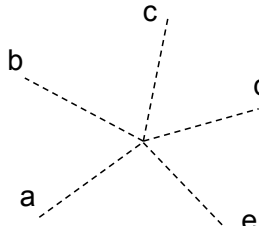
UPGMA for a dataset with n nodes requires $n - 3$ iterations. At each iteration, a matrix D with size $n \times n$ is constructed and searched. As a result, the trivial computation time is $O(n^3)$. However, for certain datasets is possible to obtained $O(n^2)$ (Gronau and Moran, 2007). Although it is a fast algorithm, it does not create a realistic result. UPGMA is only suitable when the data is ultrametric, i.e., the distances from the root to the terminal nodes are equal. Real sequence data is almost never ultrametric. Therefore, this method is not commonly applied in phylogenetics for inferring relationships (Stamatakis, 2004).

6.3.2 Neighbor Joining

Neighbor-joining methods (Saitou and Nei, 1987) use a distance matrix for tree construction similar to UPGMA. Consider n sequences and a distance matrix D . The goal is to build a tree such that the distance measured between terminal nodes (OUTs) i and j corresponds to $D[i, j]$. Fitting the exact distances in order to obtain a unique tree requires some conditions on distance values that hardly hold. Therefore, the algorithm initially sums up individual distances to calculate the divergence of an OUT (sequence) from all others and then based on this sum, a corrected distance matrix is calculated (Stamatakis, 2004). Figure 6.5 illustrates the main steps of this algorithm. It considers a tree topology such as a star and repeats the steps until the final topology of the tree with branch lengths are specified.

Example 6.2. Consider the same sequences (a, b, c, d, e) from Example 6.1 and the following star tree:

	a	b	c	d	e
a	0	5	4	7	6
b	5	0	7	10	9
c	4	7	0	7	6
d	7	10	7	0	5
e	6	9	6	5	0



Step 1. Calculate matrix M from the distance matrix.

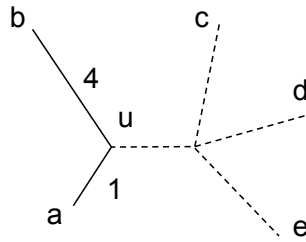
	a	b	c	d	e
a	0	-13.0	-11.5	-10.0	-10.0
b	-13.0	0	-11.5	-11.0	-11.0
c	-11.5	-11.5	0	-10.5	-10.5
d	-10.0	-11.0	-10.5	0	-13.0
e	-10.0	-11.0	-10.5	-13.0	0

Step 2. Find the minimum value in matrix M . The pairs (a, b) or (d, e) have the minimum value, -13; For this example, we choose (a, b) .

Step 3. Create a new node u and calculate the branch lengths from a and b . Then join a and b in the tree according to their lengths and the other nodes stay in the form of a star. Branches with dotted lines have unknown lengths.

$$\delta(a, u) = 5/2 + \frac{1}{2(5-2)} [(5+4+7+6) - (5+7+10+9)] = 1$$

$$\delta(b, u) = 5 - 1 = 4$$



Step 4. Create a new distance matrix D_2 from D_1 by removing the rows and columns related to a and b and add a new one for u . Calculate the distances of all other nodes to u according to the formula in Step 4.

	u	c	d	e
$D_2[u, c]$	0	3	6	5
$D_2[u, d]$	3	0	7	6
$D_2[u, e]$	6	7	0	5
	5	6	5	0

Repeat all the steps until the tree topology is fully resolved.

Neighbor-joining on a set of n nodes requires $n - 3$ iterations. At each iteration, a matrix M with size $n \times n$ is constructed and searched. As a result, the time complexity of the algorithm is $O(n^3)$. Note that this method is based on the criterion of “minimum-evolution”, i.e., at each stage, a topology is chosen so that the length of the branches has the lowest value (Gascuel and Steel, 2006). Therefore, if the distance matrix is tree-like, i.e., there is a tree that exactly corresponds to the distance matrix, then neighbor-joining returns that tree; otherwise it may not give a correct answer. Nevertheless, since Neighbor-joining is fast and does not need ultrametric data, it became the most widely used method for constructing phylogenetic trees with large data sets (Gascuel and Steel, 2006).

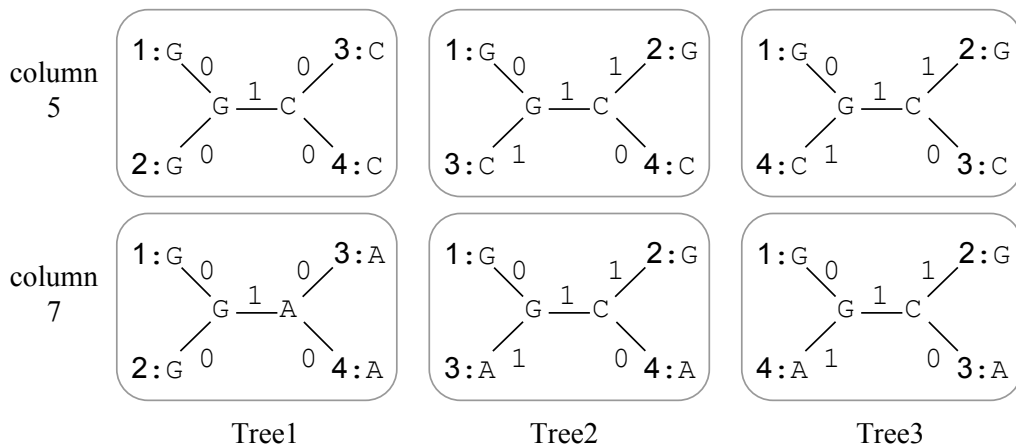
6.3.3 Maximum Parsimony

The Maximum Parsimony (MP) (Fitch, 1971) method computes many trees from the given data set and assigns a cost to each tree. MP assumes that the simplest tree is the most

	Position (column):						
Sequences:	1	2	3	4	5	6	7
1	A	G	G	A	G	T	G
2	A	A	C	A	G	T	G
3	A	G	A	A	C	T	A
4	A	G	T	A	C	T	A

Non-informative sites
Informative sites

Figure 6.6: Informative and non-informative sites in maximum parsimony method.



Number of substitutions in each columns:

	Column 5	Column 7	Total
Tree 1	1	1	2
Tree 2	3	3	6
Tree 3	3	3	6

Most parsimonious tree is Tree 1

Figure 6.7: Principles of tree construction by the maximum parsimony method. Tree 1 is the most parsimonious tree because its topology is based on the minimum number of substitutions.

plausible one. This tree is the one that requires the fewest number of substitutions to explain the data directly in the alignment. Therefore, parsimony uses the data and does not attempt to use any model to estimate the total number of substitutions. If more than one tree with the smallest number of substitutions can be obtained, then the trees are equally parsimonious. In this method, the trees are constructed by using informative sites. A site (i.e., alignment column) is considered to be informative if it has at least two different kinds of residues represented in at least two of the sequences. Figure 6.6 shows an alignment and the corresponding informative/non-informative sites. Columns 5 and 7 are informative sites. The main steps of this method are as follows:

- i) *Identify all informative sites in the multiple alignment.*
- ii) *At each informative site and for each possible tree calculate the minimum number of substitutions.*
- iii) *Sum up the number of substitutions for each possible tree.*
- iv) *A tree with the smallest number of substitutions is selected as the most likely tree.*

Figure 6.7 shows an example of tree construction by maximum parsimony using the informative sites (columns 5 and 7) from the alignment in Figure 6.6. There are three different topology tree for each column. Tree 1 in columns 5 or 7 has the minimum number of substitution, which is one. The total number of substitutions for Tree 1 is two. Therefore, Tree 1 is the most parsimonious tree since its topology is based on the minimum number of substitutions.

In MP, the search for an optimal tree is a computationally intractable problem; The large parsimony problem is challenging and is an NP-complete problem (Foulds and Graham, 1982). However, a tree can be computed in polynomial time. Given n aligned sequences of length k , the most parsimonious tree can be computed in $O(nk)$ (Fitch, 1971). Furthermore, the MP works well when the sequences being compared are not too divergent. However, even if the sequences are not very different, but the *long branch attraction scenario* arises, then the resulting tree maybe misleading (Hendy and Penny, 1989; Huelsenbeck, 1995).

6.3.4 Maximum Likelihood

Maximum likelihood (ML) (Felsenstein, 1981) is a statistical method that estimates the unknown parameters of a probability model. The ML method is currently widely used for the construction of the phylogenetic tree. ML evaluates the probability that the selected evolutionary model predicts the observed sequences. In other words, the topology of the phylogenetic trees constructed using maximum likelihood has the highest probability of producing the observed sequences.

It is known that ML is a consistent method. Moreover, it has the ability to make statistical comparisons between topologies and data sets, and it can return several equally likely trees. A disadvantage of ML is the extensive computation (Egan and Crandall, 2006). Finding optimal ML trees in general is an NP-hard problem (Chor and Tuller, 2006). Exact algorithms are usually bounded to 20 sequences. For larger sequences, heuristic approaches are commonly applied.

6.3.5 Overall Discussion

Distance-based methods require evolutionary distance, i.e., the number of substitutions that have occurred along the branches between two sequences, between all pairs of OUTs. By summarizing the input data (e.g., sequence) into a matrix of pairwise distances, distance methods inevitably face a loss of information; this usually leads to less accurate tree reconstructions than those of character-based methods, which entirely use the input data.

Due to the simplicity of their input, heuristic distance methods are the fastest available and, therefore, are preferable when speed is an essential requirement (e.g., a large number of taxa, or when a large number of trees must be generated). Moreover, they also produce only one tree, and thus it is not possible to examine competing hypotheses about the relationship between sequences.

On the contrary, character-based methods operate directly on the aligned sequences rather than on pairwise distances. MP does not require an explicit model of sequence evolution; it identifies the trees that involves the smallest number of mutational substitutions necessary to explain the differences among the data at hand. In many cases, MP methods are relatively free of assumptions considering nucleotide and amino acid substitution. However, these methods can lead to wrong relationship estimates for similar sequences (Hendy and Penny, 1989; Huelsenbeck, 1995). In ML, the topology that gives the highest ML value is chosen as the final tree. One of the strengths of the ML method is the ease with which

hypotheses about evolutionary relationships can be formulated. However, this method is computationally very intensive, and it cannot be used for very large datasets.

The distance-based methods are based on distances obtained from a single multiple sequence alignment. However, different alignments may lead to different tree topologies (Wong et al., 2008). Moreover, statistical methods such as ML may be statistically inconsistent with respect to a general model of sequence evolution, since they treat indels as missing data (Warnow, 2012). We believe that by considering a set of Pareto optimal alignments instead of a single alignment, different tree topologies can be obtained, from which a practitioner can better infer the relationship between the species under analysis.

In the next section, we present a new method that directly reconstructs phylogenetic trees from the non-dominated score set data. We use the algorithms introduced in Chapter 4 to compute the non-dominated score set for each pair of sequences under study. Then, from this data, we construct several tree topologies. Furthermore, we introduce a new procedure based on bootstrapping to validate the reliability of trees.

6.4 A Biobjective Method for Constructing Phylogenetic Trees

In this section, we describe a new method for constructing phylogenetic trees from the non-dominated score set. Consider n sequences with a collection of non-dominated score sets each of which obtained from each pair of sequences. For each gap/indel value that arises from the union of all non-dominated scores sets, we build a phylogenetic tree. The algorithm consists of the following four steps for each gap/indel value d :

1. *Collect all the substitution score values of score vectors that have d indels/gaps.*
2. *Normalize each substitution score value between 0 and 1.*
3. *Compute the distance between two sequences as one minus the normalized substitution score value of the non-dominated score vector obtained from those two sequences.*
4. *Build a phylogenetic tree using the distances computed above and using Neighbor-Joining method¹.*

An essential aspect of the analysis is to understand how often specific phylogenetic tree topologies arise from this approach (see Section 6.2). A less frequent topology, or tree branch,

¹In this step the PHYLIP package (Felsenstein, 1985) is used in our experiments.

may indicate a less reliable relation between the corresponding species. Therefore, we count how many times each branch arises in all phylogenetic trees and add the corresponding relative frequency to each branch. Note that this information is analogous to the bootstrap values (Felsenstein, 1985); instead of the sampling process, we use the non-dominated score sets.

6.5 Analysis with Real Data

In this section, two data sets are used for illustration purpose. The results trees are compared with those obtained with the well-known technique of Maximum Likelihood.

6.5.1 First Experiment

This experiment consists of dataset *Candida* genes, *C.albicans PAPA*, *C.albicans PAPalpha*, *C.tropicalis PAPalpha*, *C.tropicalis PAPA* and *C.dublinsiensis PAPA*, as well as the genes *Pichia stipitis PAPA*, and *Saccharomyces cerevisiae PAP* (See Butler et al., 2009 for more information about these genes). For each pair of genes, we computed the non-dominated score set with respect to Problem (VSGP) with substitution matrix $\mathbf{M}[i, i] := 1$ and $\mathbf{M}[i, j] := -1$, $i \neq j$; see the complete non-dominated score sets with staircase line representation in Figure 6.9. It is possible to observe that there exists a large number of non-extreme supported score vectors, as indicated by the straight lines.

We constructed 567 phylogenetic trees by using our method. Although we have obtained 567 phylogenetic trees, only two different topologies were obtained. One of them was discarded from the analysis since it arose only once. Figure 6.8 (a) shows the remaining tree obtained for a gap value of 283, the median of all gap values found; the value close to each branch indicates the relative frequency of that branch. For comparison, we computed the evolutionary tree using the Maximum Likelihood method based on the model from Felsenstein (1985) obtained by MEGA5 (Tamura et al., 2011); see Figure 6.8 (b). The bootstrap consensus tree inferred from 1000 replicates was taken to represent the evolutionary history of the taxonomic analysis. There were a total of 1613 positions in the final dataset with all of them having less than 90% site coverage removed. The percentage of trees in which the associated taxa clustered together is shown next to the branches. Initial tree(s) for the heuristic search were obtained automatically as follows: when the number of common sites was less than 100 or less than one-fourth of the total number of sites, the maximum parsimony (Fitch, 1971) method was used; otherwise, BIONJ method (Gascuel, 1997) with MCL

distance matrix was used. The tree is drawn to scale, with branch lengths measured in the number of substitutions per site.

By comparing both trees, it is possible to infer that they have a similar topology and similar relative branch frequency. This conclusion can also be understood by the regularity of the lines of Figure 6.9. The nearest genes are *C.albicans PAPalpha* and *C.dublinsiensis PAPA*, which corresponds to the black line in Figure 6.9, with the largest substitution score. In the same figure, the three pink lines allow us to infer the positioning of *C.tropicalis PAPalpha* with respect to *C.albicans PAPalpha* and *C.dublinsiensis PAPA*, as well as the branch (*C.albicansPAPA, C.dublinsiensis PAPA*). As expected, *P. stipitis PAP* and *S. Cerevisiae PAP* are the most distant species, as seen by the green dot-dash line in Figure 6.9.

Finally, note that both phylogenetic trees indicate a lower value for the branch that contains *P. stipitis PAP* gene. We relate this value to the crossed lines between the pairs formed by *P. stipitis PAP* and other genes (see blue lines in Figure 6.9).

6.5.2 Second Experiment

The second data set is a classic example of comparison between primates: *Homo sapiens haplogroup J1c3*, *Homo sapiens neanderthalensis*, *Gorilla gorilla graueri*, *Pan troglodytes troglodytes* and *Pongo abelii* species. We performed the same analysis as described in the previous section. Figure 6.11 shows the non-dominated score sets with staircase line representation. The figure shows a large number of non-extreme supported score vectors and some intersecting lines. This indicates that the relationship between those species, in the context of evolutionary studies, may depend on the score vectors chosen. We conjecture that the existence of intersections may indicate a less reliable conclusion about the evolutionary relationship.

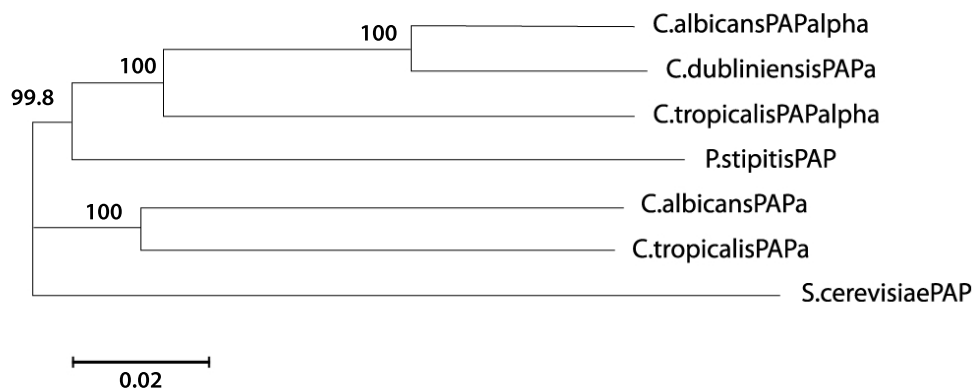
By employing our biobjective method, a total of 144 phylogenetic trees were obtained, which gave rise to two different tree topologies. Plots (a) and (b) of Figure 6.10 show the two trees topologies obtained for a gap value of 22 and 54, respectively; these gaps values correspond to the median of all gap values found in the phylogenetic trees with the same topology. The evolutionary tree using the ML method was also computed using the same method described in the previous section. The tree with the highest log likelihood (-1351.1512) is shown in Figure 6.10 (c). All positions with less than 60% site coverage were eliminated. That is, fewer than 40% alignment gaps, missing data, and ambiguous bases were allowed at any position. There were a total of 376 positions in the final dataset.

The two trees obtained with our method differ slightly in the relationship of the *Pan*

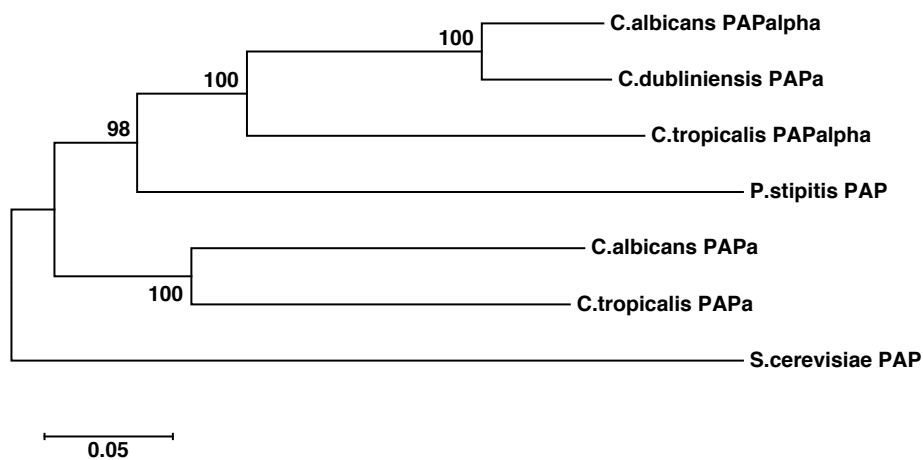
traglodytes traglodytes with the remaining species: in plot (a), this species arose in the top of the branch (*Homo sapiens neanderthalensis*, *Homo sapiens haplogrop J1c3*), whereas in plot (b), it arises in a different clade, paired with *Gorilla gorilla graueri*. The relative branch frequencies suggest that the tree of the plot (b) may be more reliable. Interestingly, this is also confirmed by the tree obtained with the ML method.

6.6 Summary and Discussion

In this chapter, we showed for the first time a successful link between non-dominated score sets and phylogenetic tree construction. We propose a method based on our biobjective framework that allows to construct phylogenetic trees as well as to give information about the reliability of the tree branches. The advantage of this method is that no assumption about a priori knowledge of users preferences is required; therefore, being less biased. We evaluated this method with two real-life benchmark datasets. Although two many trees have resulted, only a few topologies of interest were obtained and matched those obtained with the maximum likelihood method. On the first input, the produced trees were quite similar, on the second one, one of the two was similar, and the other one was different. For the future work, further research is needed to derive a theoretical relation between maximum likelihood estimations and the information provided by the non-dominated score set for tree branch reliability.



(a)



(b)

Figure 6.8: Phylogenetic trees for the first experiment: biobjective model for a gap value of 283 (a) and Maximum Likelihood model (b).

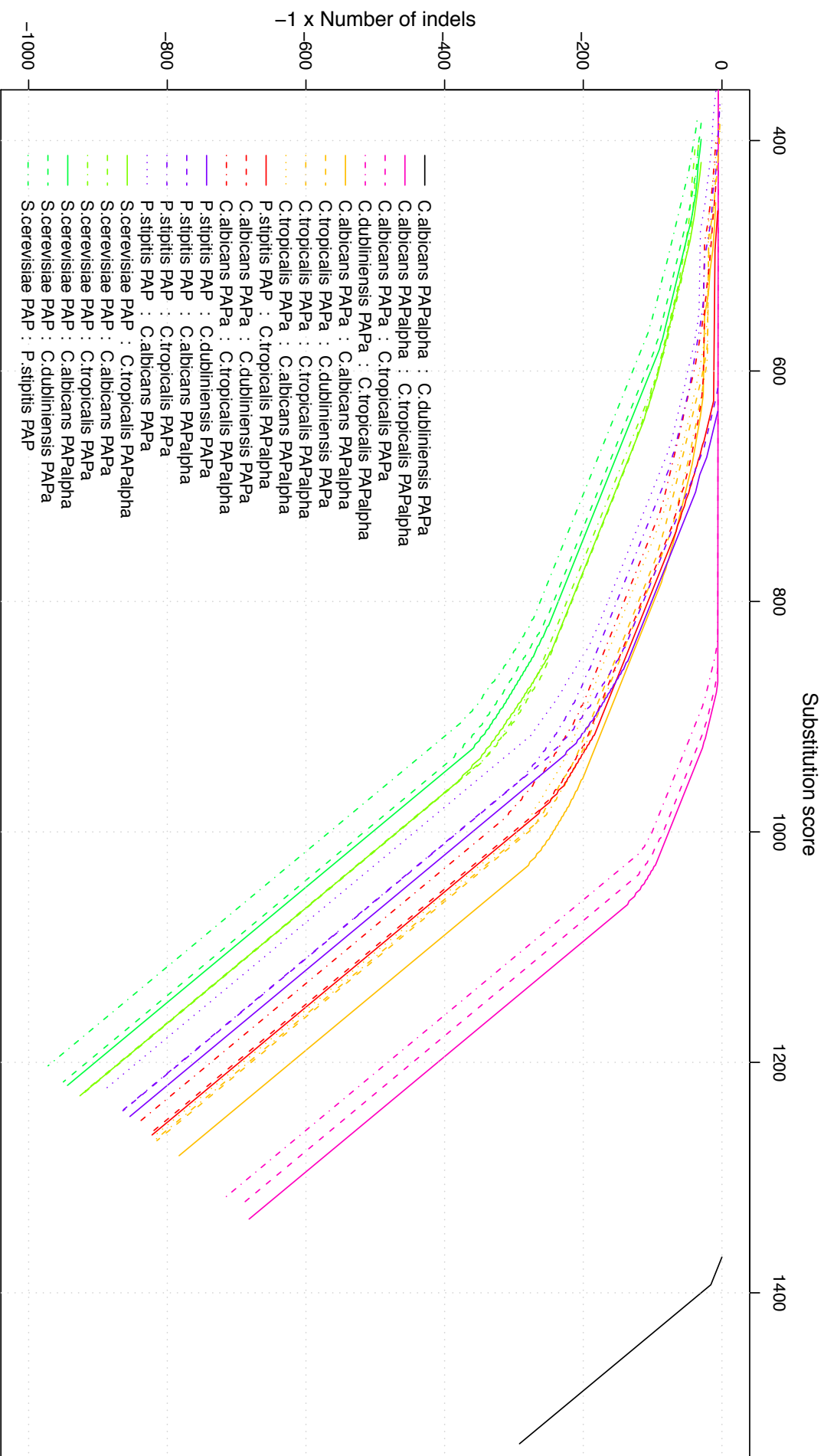


Figure 9.9: Staircase line representation of the non-dominated score sets for the first experiment.

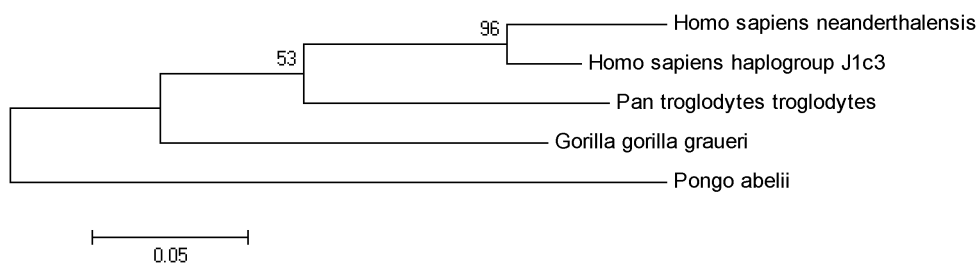
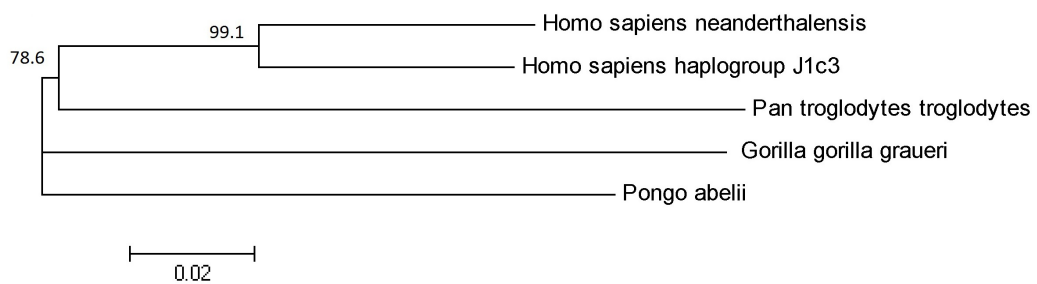
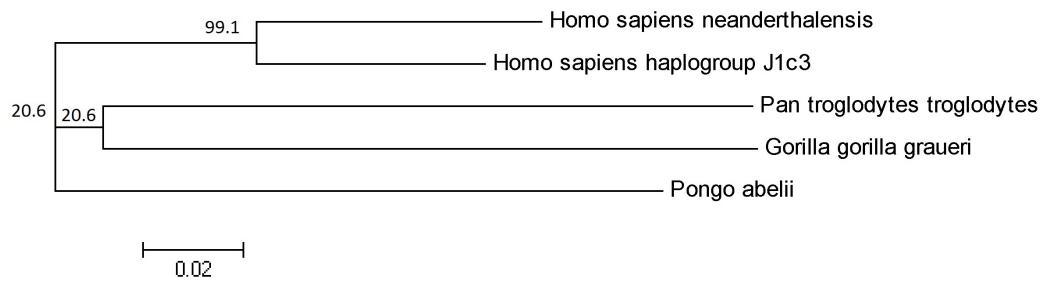


Figure 6.10: Phylogenetic trees for the second experiment: biobjective model for a gap value of 22 (a) and 54 (b) and Maximum Likelihood model (c).

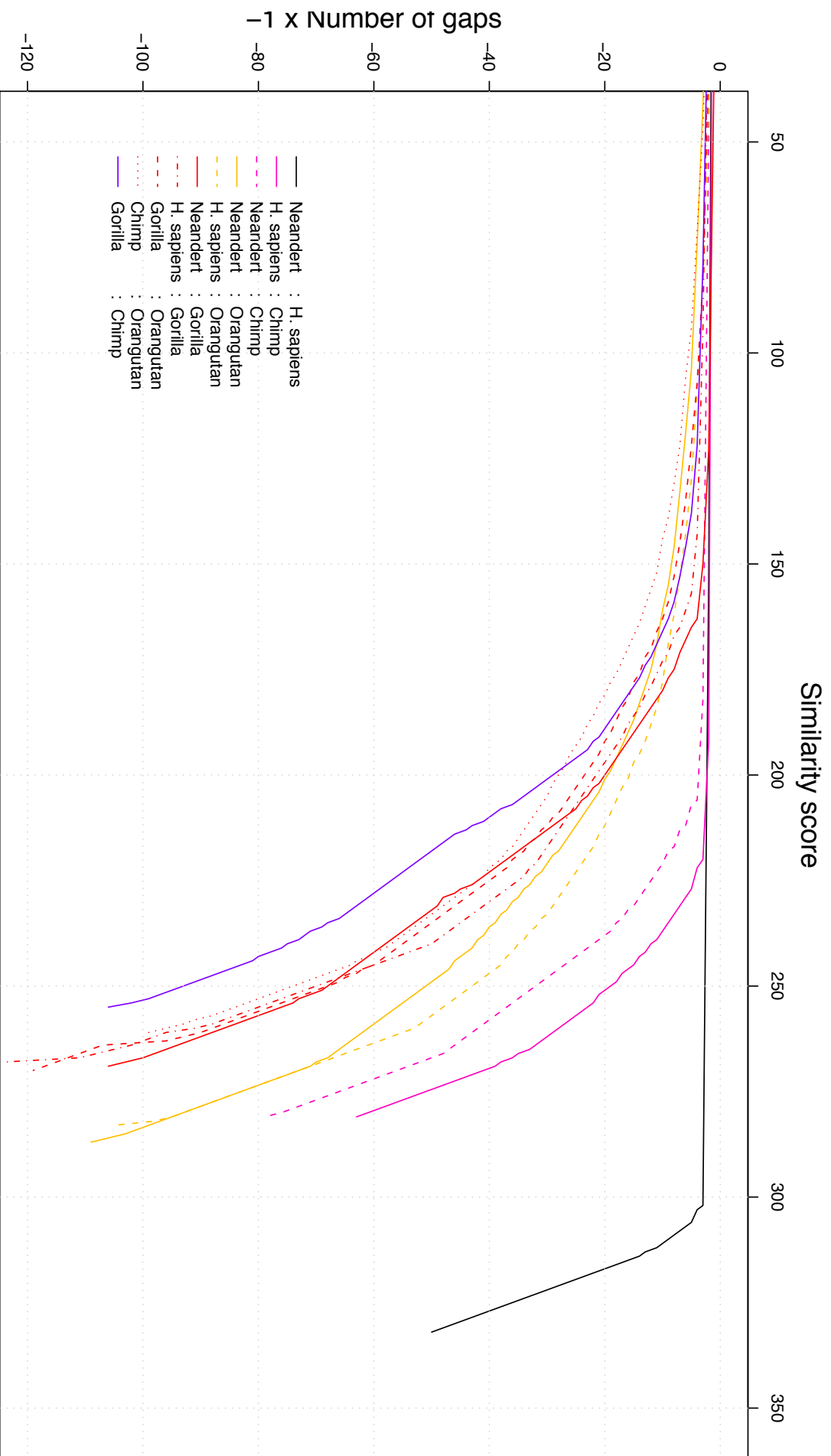


Figure 6.11: Staircase line representation of the non-dominated score sets for the second experiment.

Chapter 7

Conclusions and Future Work

7.1 Summary and Contributions

A practitioner of Bioinformatics needs to use programs that solve particular optimization problems in order to analyze the biological data at hand. From our point of view, these problems are naturally multiobjective and deal with conflicting objectives. The current approaches oversimplify this problem by assuming specific preference information. However, this preference information is not widely accepted by the community and cannot be easily defined by a practitioner for a given data.

In this thesis, we have studied how multiobjective concepts can be used for sequence alignment, a core problem in Bioinformatics. We have presented different multiobjective formulations of the sequence alignment problem as well as several ways of tackling them. Our findings show that the multiobjective pairwise sequence alignment problem can be solved efficiently to optimality by extending principles of dynamic programming algorithms since the size of the set of optimal scores is linear with respect to the problem size. Noteworthy, this is a particular feature that does not arise on multiobjective combinatorial optimization problems, in general. Moreover, we have introduced effective speed-up techniques that also use fewer memory resources. Besides, we have adapted the classical principles of ϵ -constraint to solve the same problems, but the experimental results suggest that this approach requires

more computational time.

We have also considered the multiobjective multiple sequence alignment, a natural extension of pairwise sequence alignment for more than two sequences. Given that this problem is NP-hard for an arbitrary number of sequences, we have tackled it with heuristic methods based on local search procedures and multiobjective evolutionary algorithms. These approaches return an approximation to the optimal scores at a given time defined by the user. We conducted an in-depth experimental analysis on a wide range of benchmark instances, which allowed us to relate the impact of algorithmic parameters on performance. Moreover, we established functional relationships between instances features and algorithm performance. The experimental results suggested that a simple local search can provide high-quality results in a short amount of time.

Finally, we proposed a method that uses multiobjective concepts for the construction of phylogenetic trees and to assess their reliability. Our method was tested on two well-known real-life biological data. The results suggested that very few distinct tree topologies can be obtained, each of which gives a possible relationship between the species under analysis. This is a critical point of our thesis – the current approaches used by most practitioners can only carry one perspective over the biological data. Differently, our methods provide alternative perspectives, giving further insight to the practitioner.

In this work, we prove the correctness of our algorithms, discuss their time complexity and analyze their performance in practice on well-known benchmark data sets. The most effective dynamic programming algorithms presented in this thesis for the sequence alignment problem with two sequences are publicly available at the website <http://mosal.dei.uc.pt>, which, in addition, allows the user to run them online and visualize the set of alignments.

7.2 Future Work and Open Issues

We clearly think that much work has still to be done in the research on multiobjective algorithms for bioinformatics problems in order to meet the knowledge gathered for the single-objective case (Handl et al., 2007). In the following, we describe some future work and open issues concerned with the main topic of this thesis.

- **Extension to more objectives.** The dynamic programming algorithm for mPSA can be extended for the three objective case, where the substitution score, the number of indels and the number of gaps are simultaneously considered in the score vector function.

- **Representative set of Pareto optimal alignments.** The number of alignments returned by our approaches is still considerably large, which may be overwhelming for the practitioner. Selecting the best alignments is out of the scope of this thesis, but one could use methods discussed in the field of Multiple Criteria Decision Analysis to support the decision. One possibility is to present a smaller subset of alignments that are *representative* of the complete trade-off; such notion of representativeness may be based on metrics of uniformity (the most spread subset) or coverage (the subset that best covers the whole set). Algorithms that allow finding optimal representative subsets for the biobjective case are available in (Vaz et al., 2015).
- **Improvement in local search.** The local search introduced in this article is able to provide high-quality alignments in a reasonable amount of time. However, there are still ways of improving it further. For the future work, we might try other starting alignments that are not just based on the sequence similarities but also taking into account other biological criteria such as the secondary structure of the alignments. Further, we may test different perturbation methods. One of the possibilities is to use the phylogenetic trees of the sequences and apply the ratchet strategy (Nixon, 1999; Morrison, 2007) which may help to create new solutions with more information from existing results.
- **Phylogenetic tree construction.** Further research is needed to derive a theoretical relationship between maximum likelihood estimations and the information provided by the non-dominated score set for tree branch reliability.

We believed that this thesis covered some fundamental aspects of the design and analysis of multiobjective optimization problem for sequence alignments by providing conceptual search principles and tools for performance assessment. However, we also feel that our contribution is nothing more than a small step towards the more effective use of multiobjective algorithms for problem-solving. We think that much more has to be done, and we hope that answers to the questions posed above contribute actively to further improvement in the next years of research on this topic.

Appendix A

Experimental Results for the mMSA

In this chapter, we report the results obtained with NSGA-II and IPLS for the mMSA, with respect to Chapter 5.

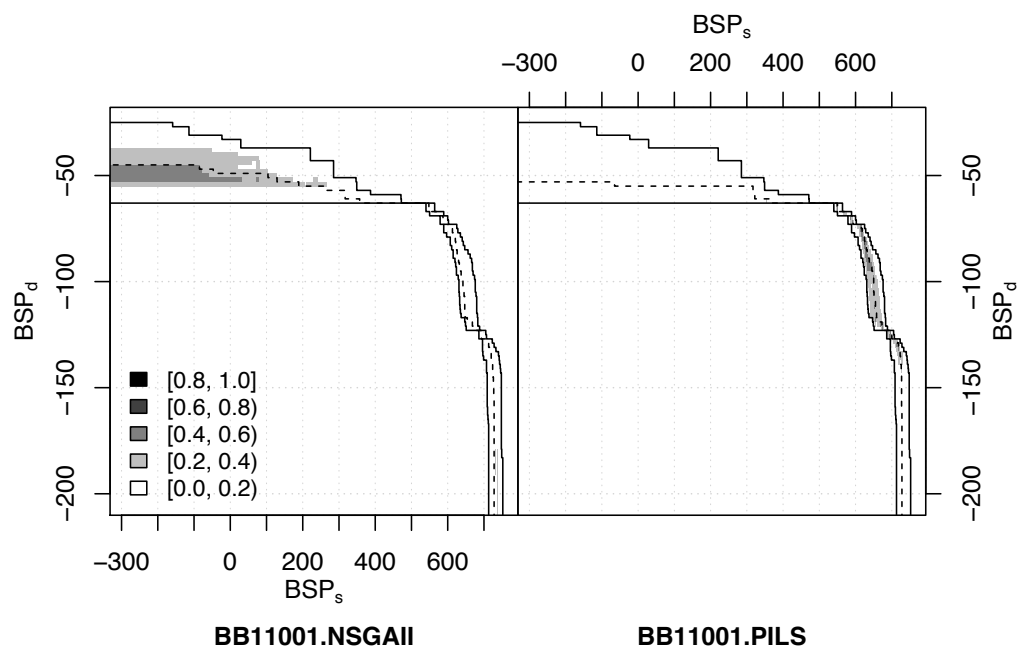


Figure A.1: Plot of the EAF for the instance BB11001

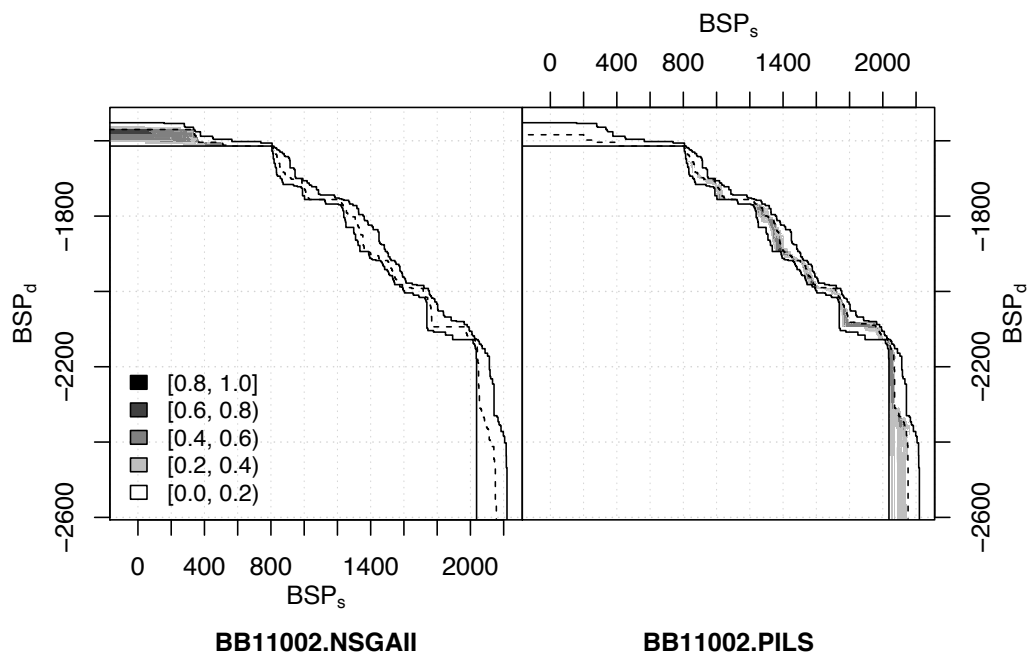


Figure A.2: Plot of the EAF for the instance BB11002

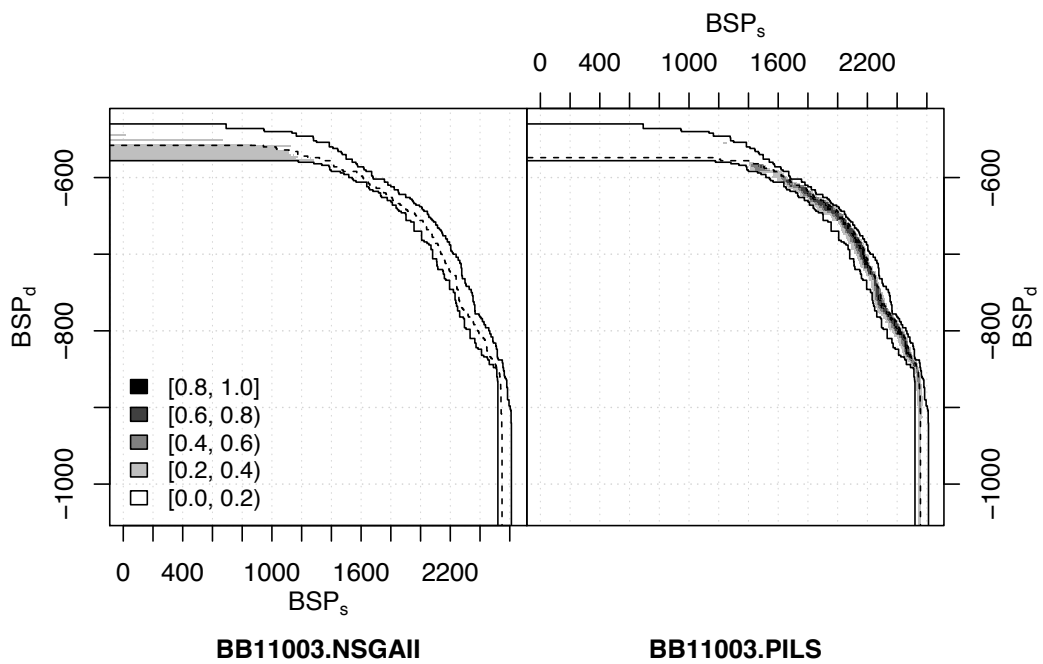


Figure A.3: Plot of the EAF for the instance BB11003

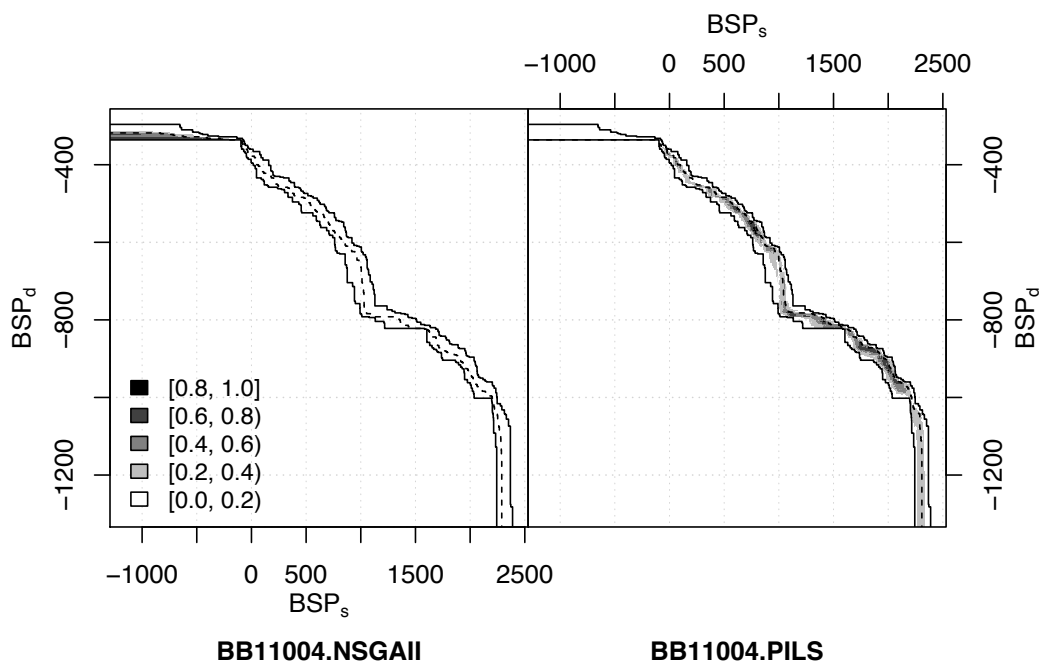


Figure A.4: Plot of the EAF for the instance BB11004

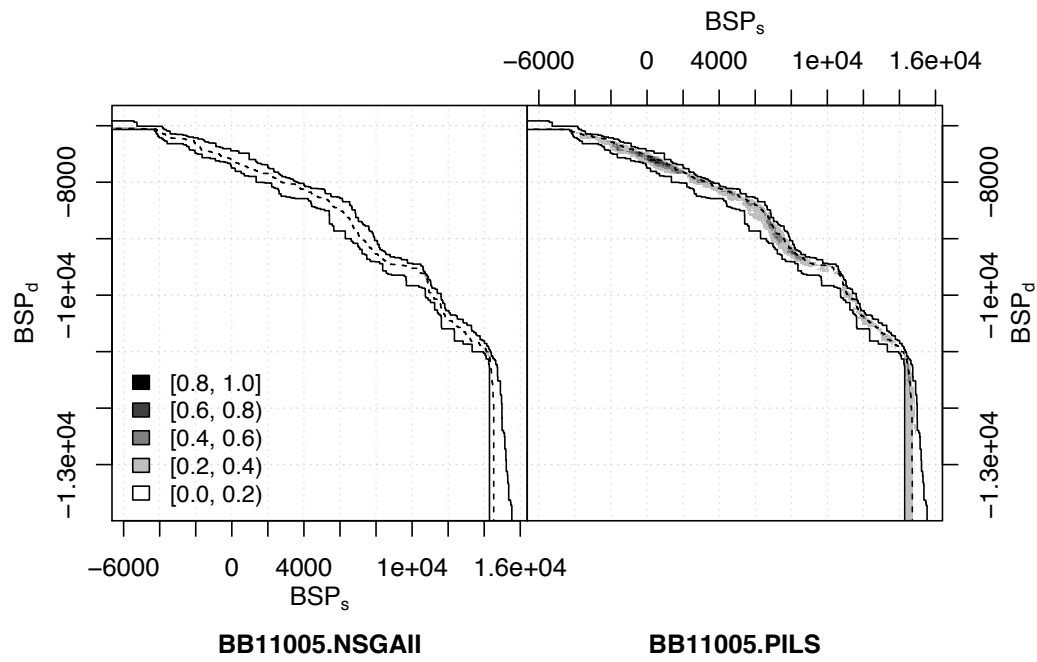


Figure A.5: Plot of the EAF for the instance BB11005

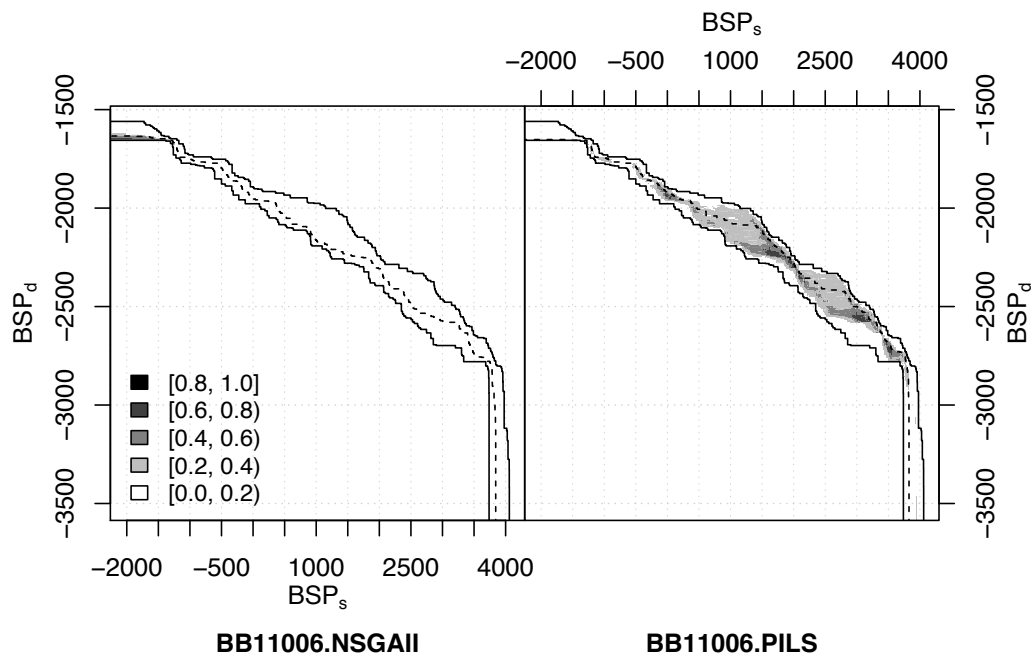


Figure A.6: Plot of the EAF for the instance BB11006

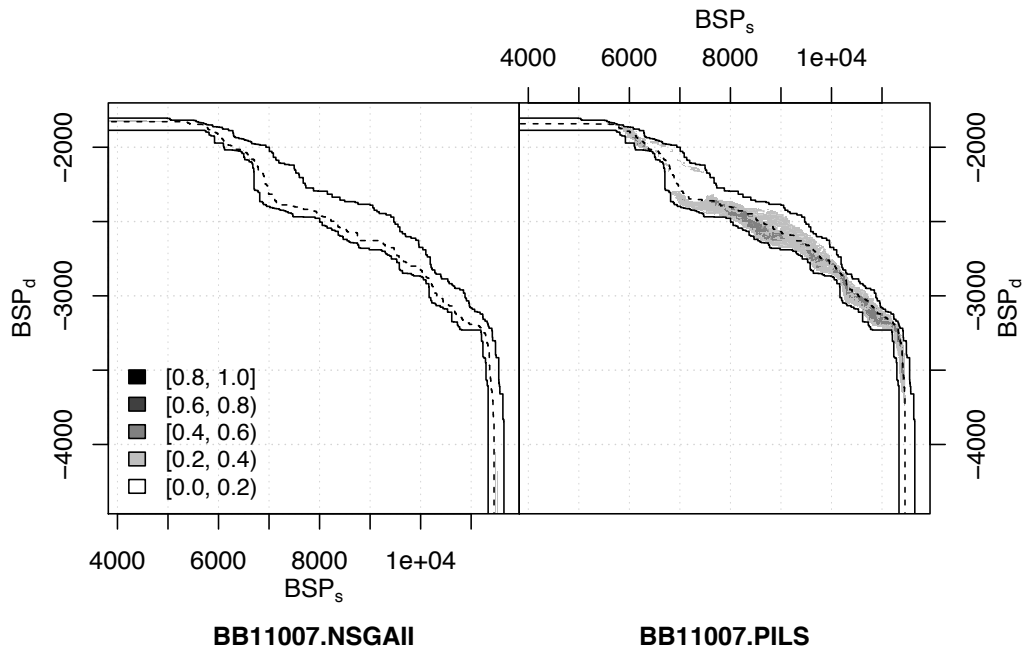


Figure A.7: Plot of the EAF for the instance BB11007

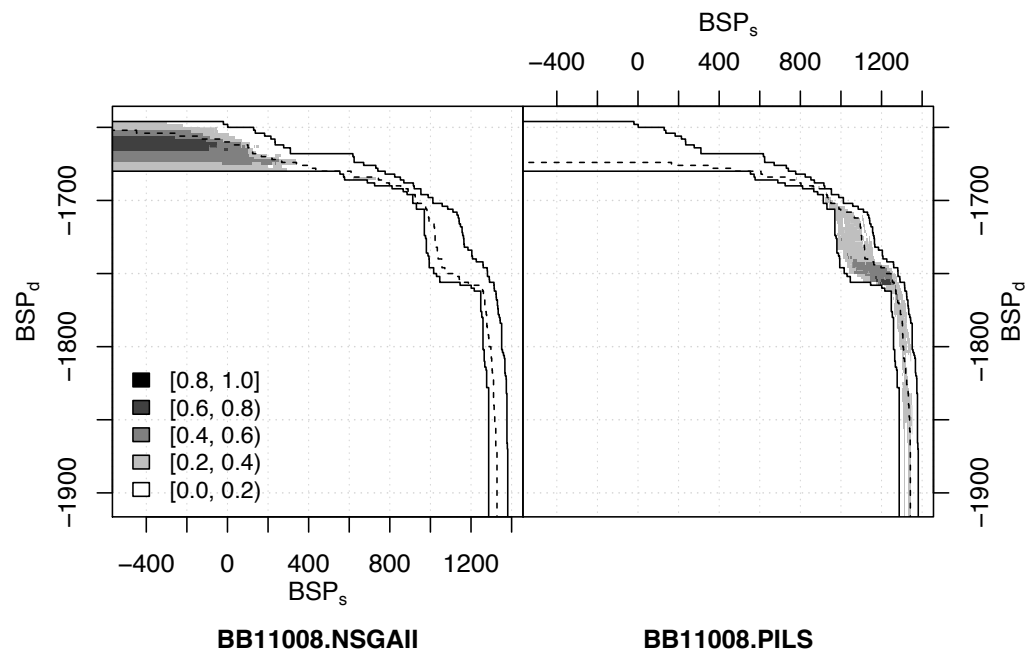


Figure A.8: Plot of the EAF for the instance BB11008

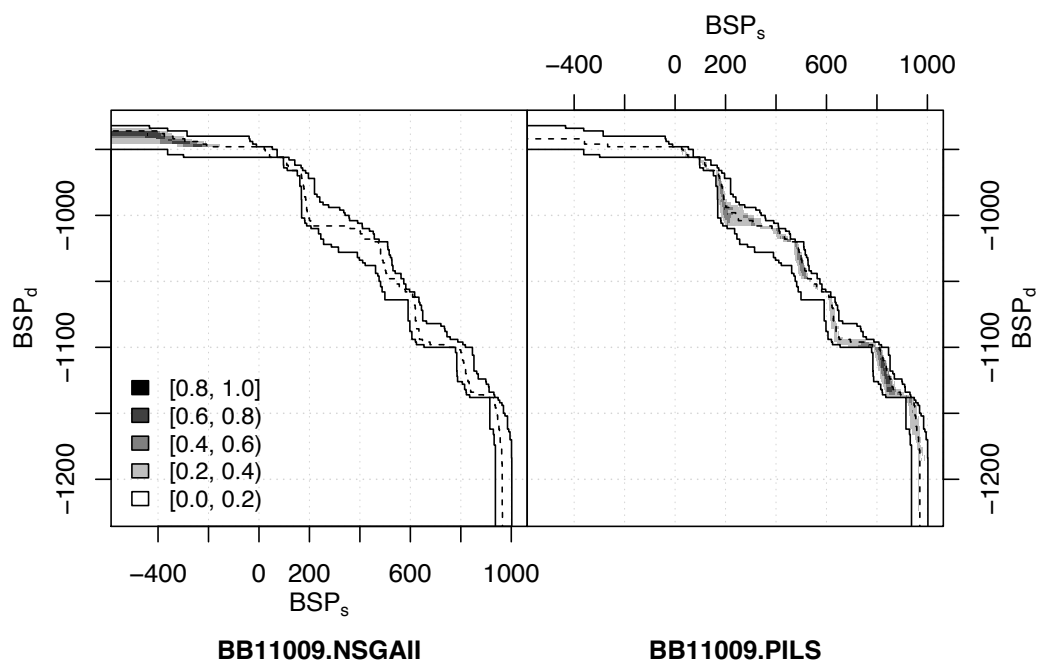


Figure A.9: Plot of the EAF for the instance BB11009

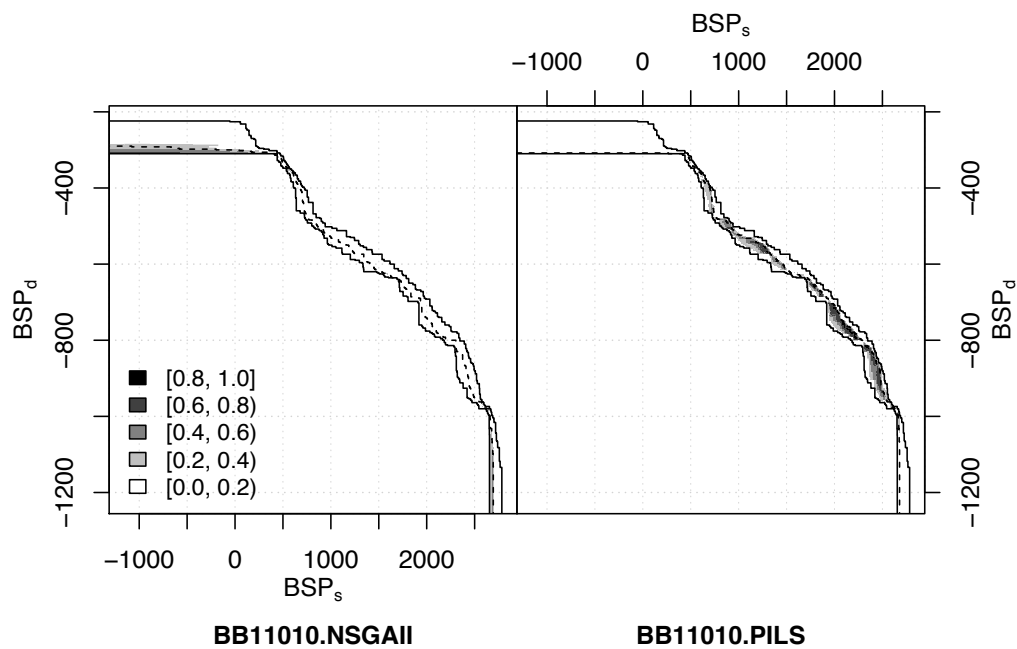


Figure A.10: Plot of the EAF for the instance BB11010

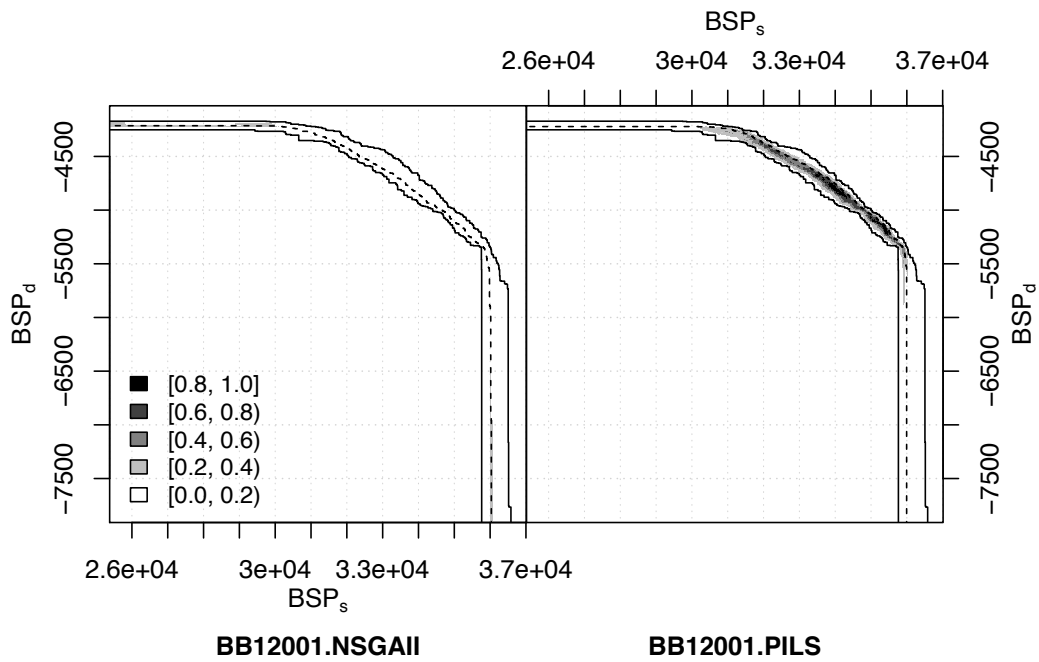


Figure A.11: Plot of the EAF for the instance BB12001

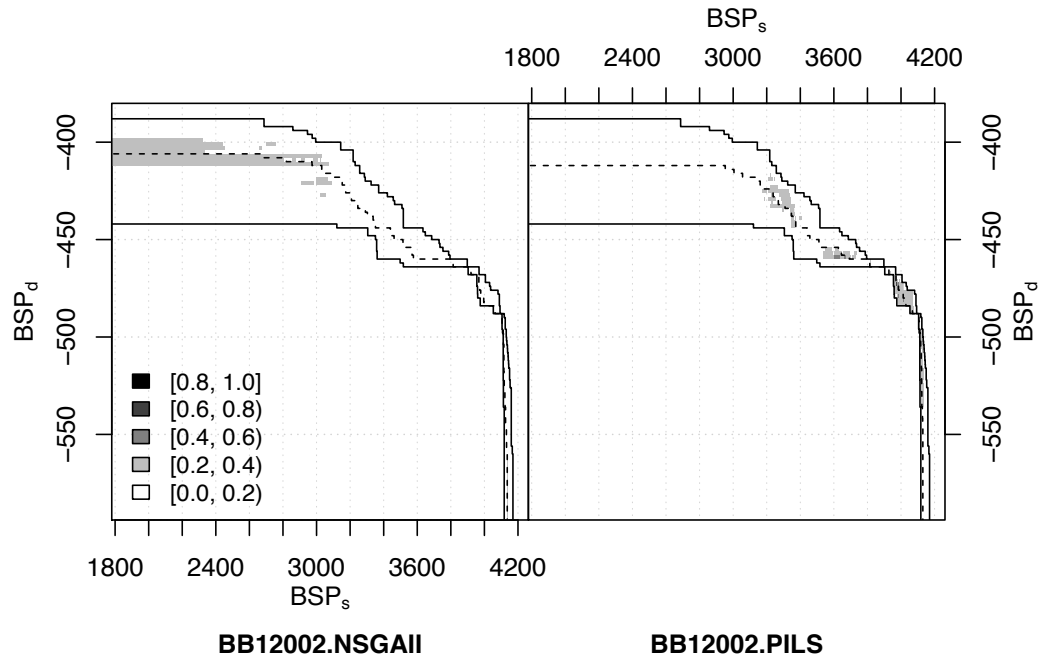


Figure A.12: Plot of the EAF for the instance BB12002

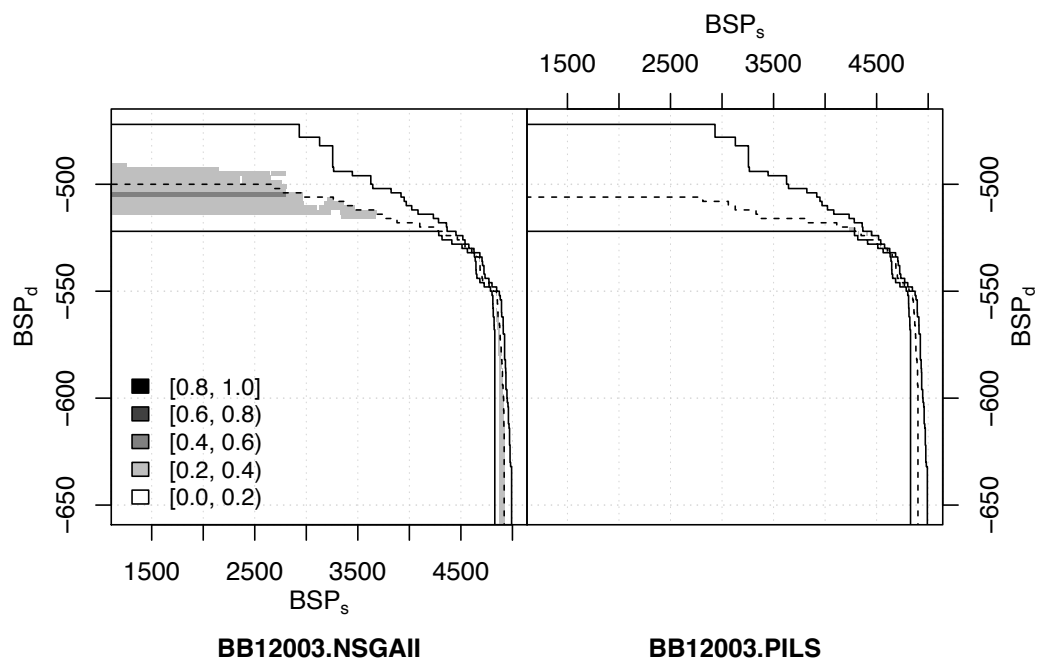


Figure A.13: Plot of the EAF for the instance BB12003

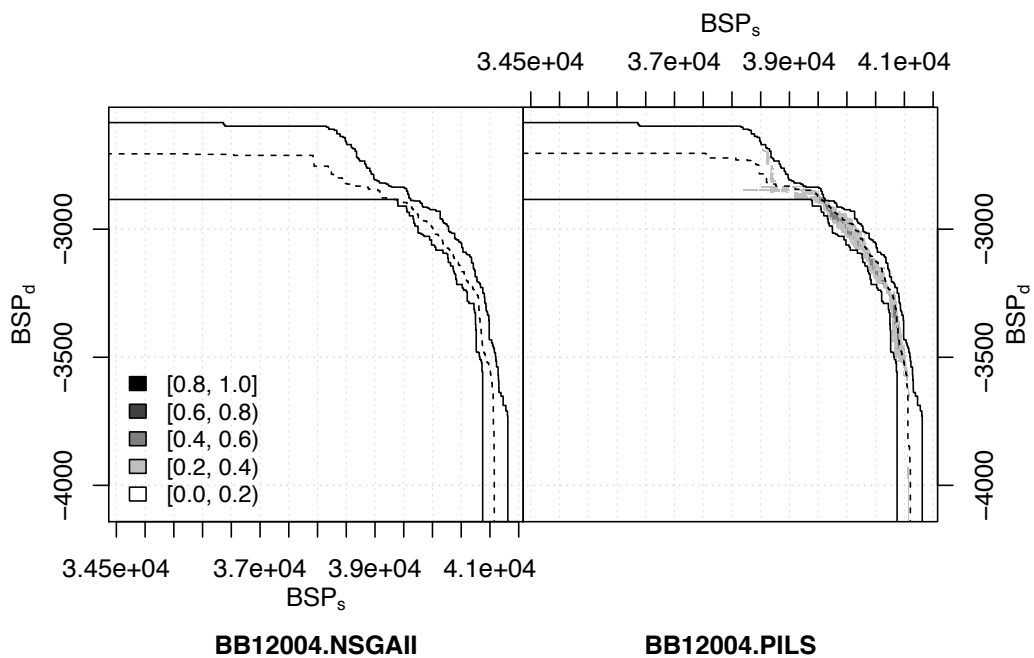


Figure A.14: Plot of the EAF for the instance BB12004

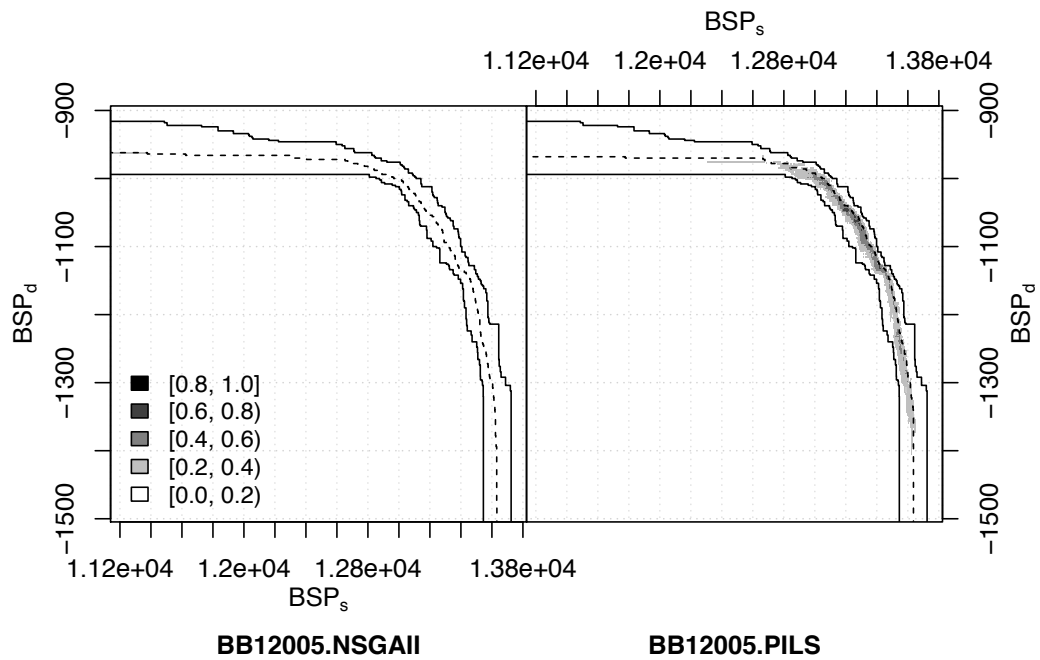


Figure A.15: Plot of the EAF for the instance BB12005

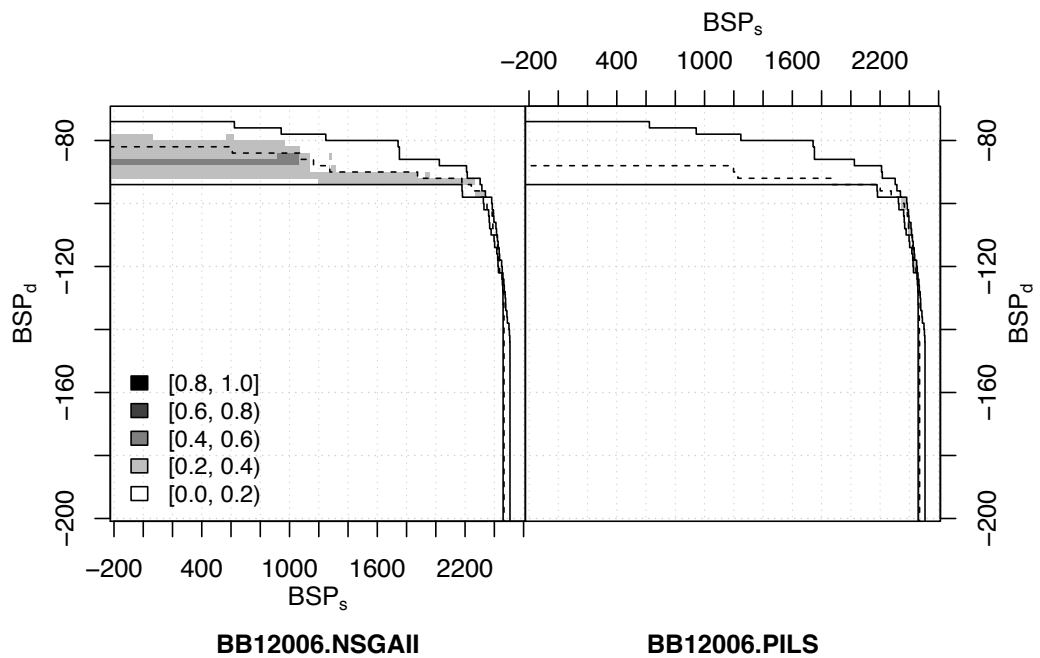


Figure A.16: Plot of the EAF for the instance BB12006

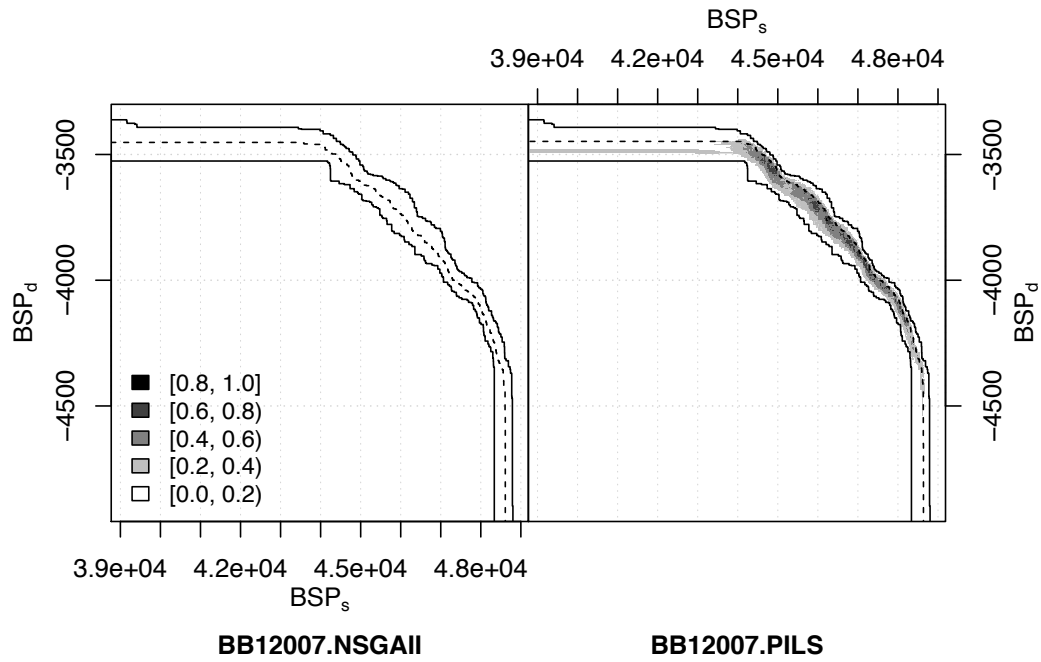


Figure A.17: Plot of the EAF for the instance BB12007

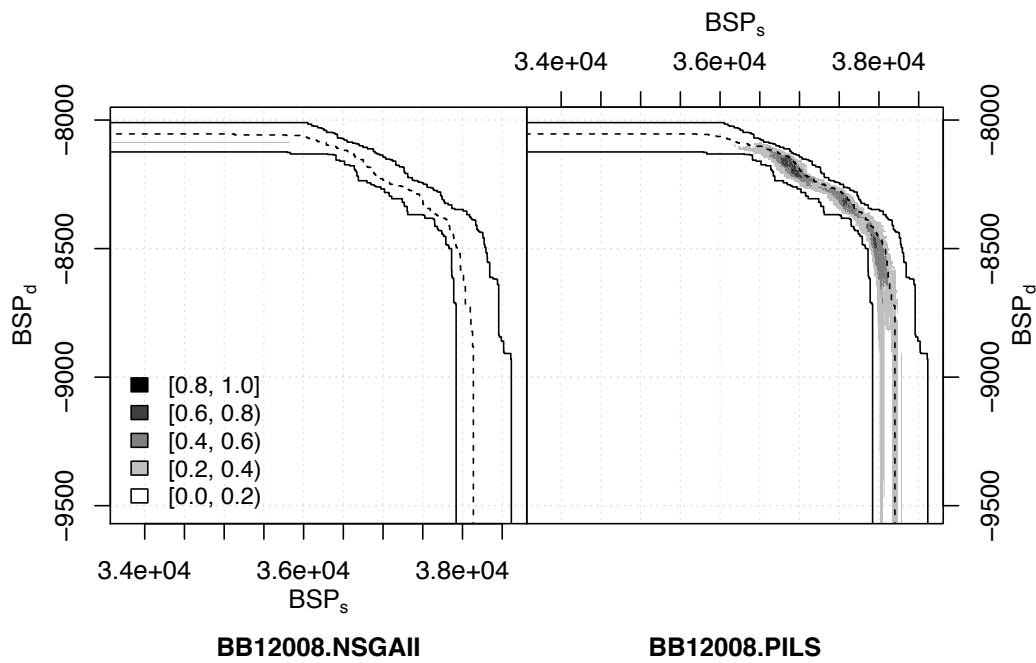


Figure A.18: Plot of the EAF for the instance BB12008

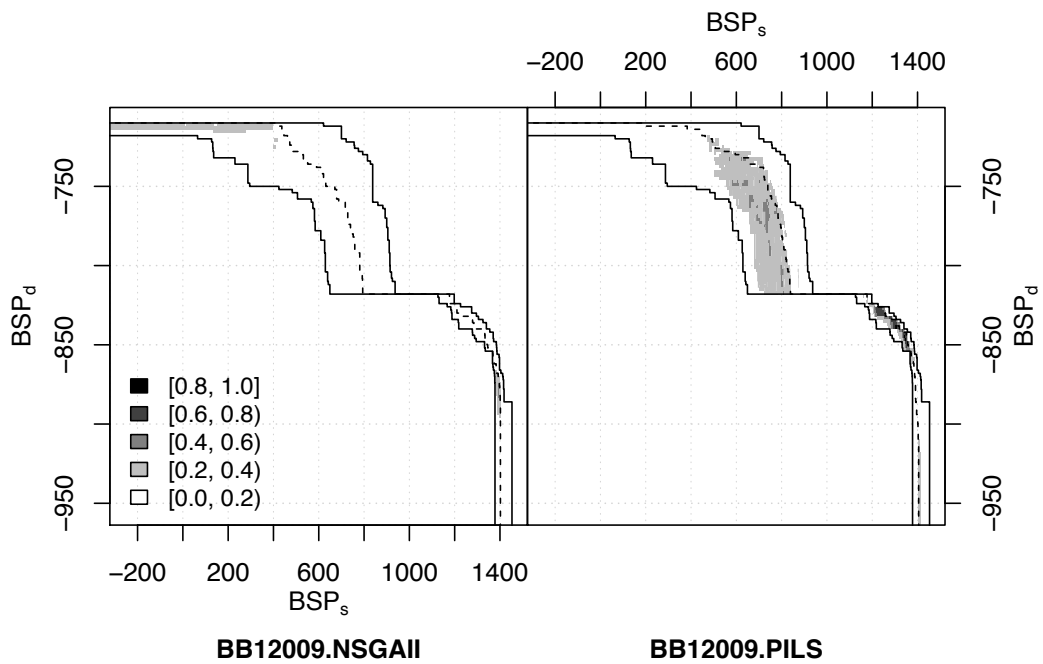


Figure A.19: Plot of the EAF for the instance BB12009

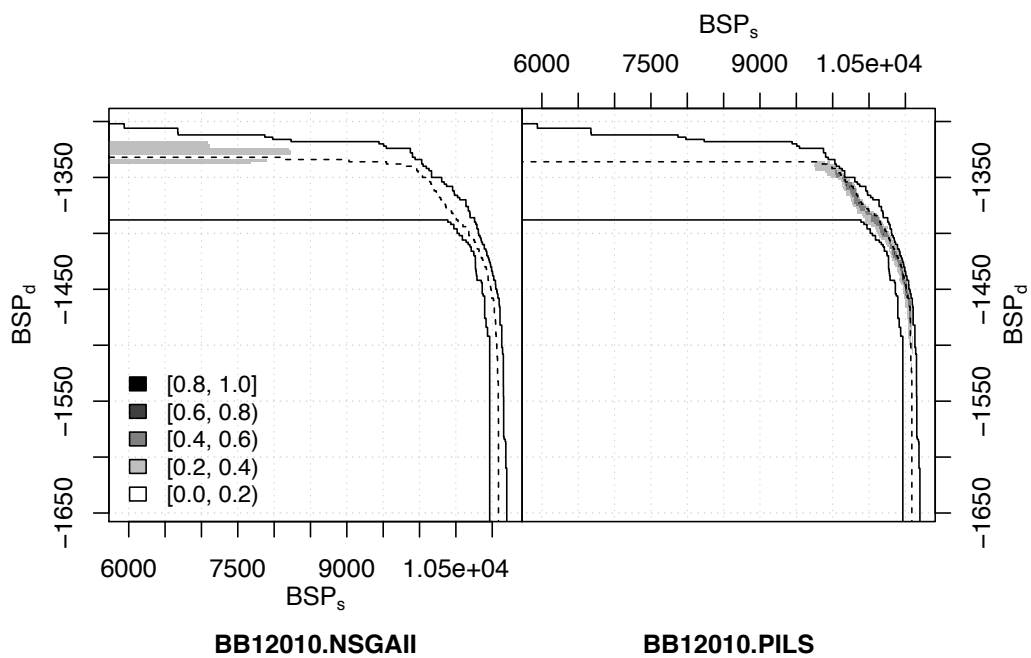


Figure A.20: Plot of the EAF for the instance BB12010

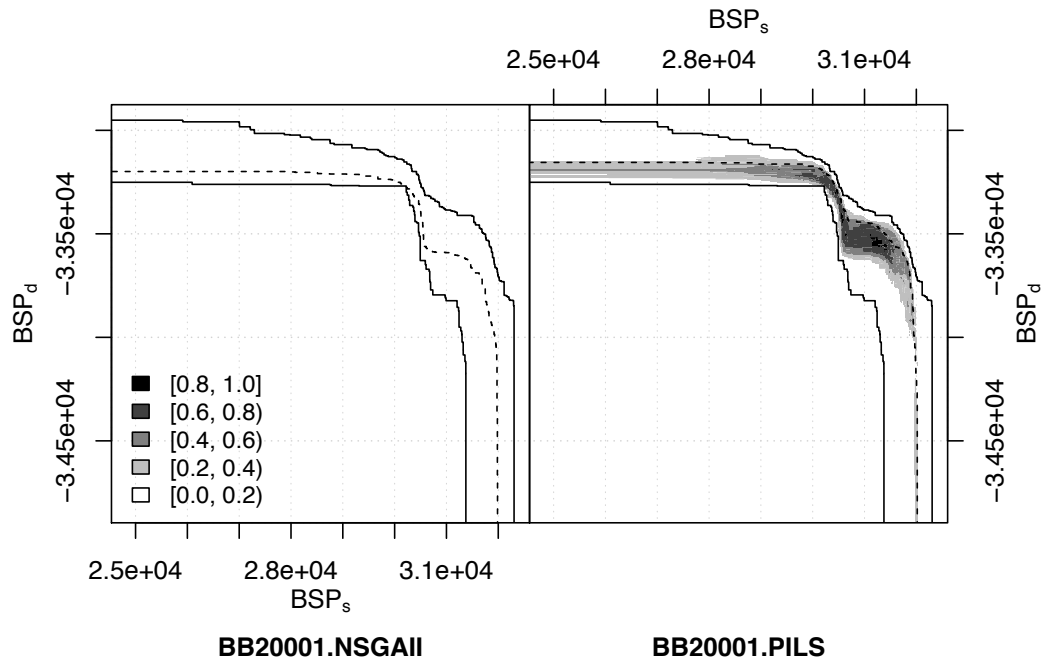


Figure A.21: Plot of the EAF for the instance BB20001

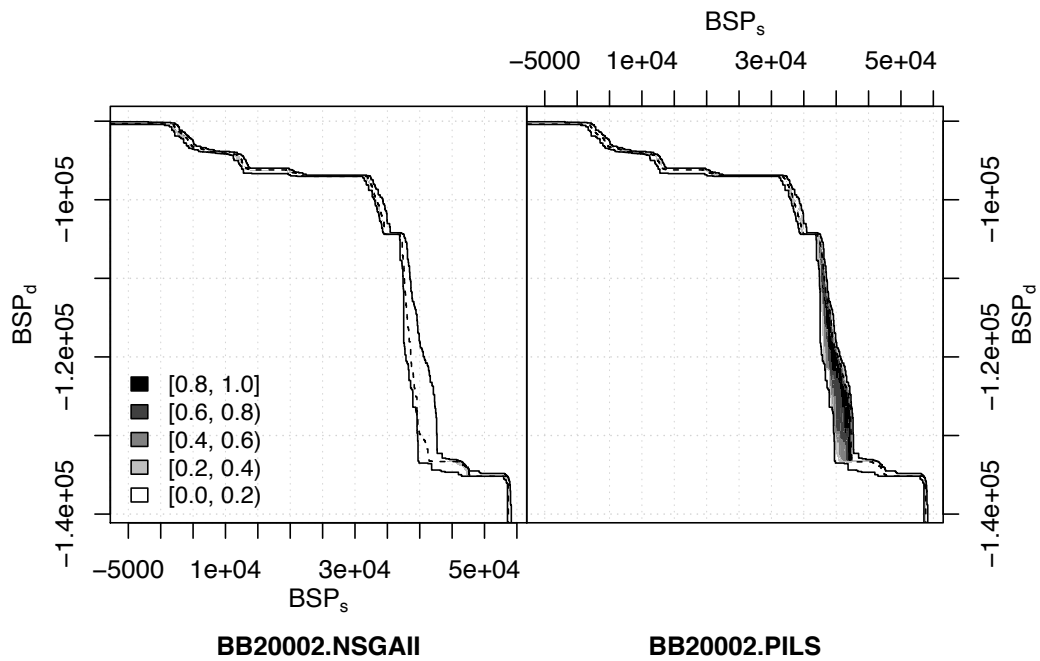


Figure A.22: Plot of the EAF for the instance BB20002

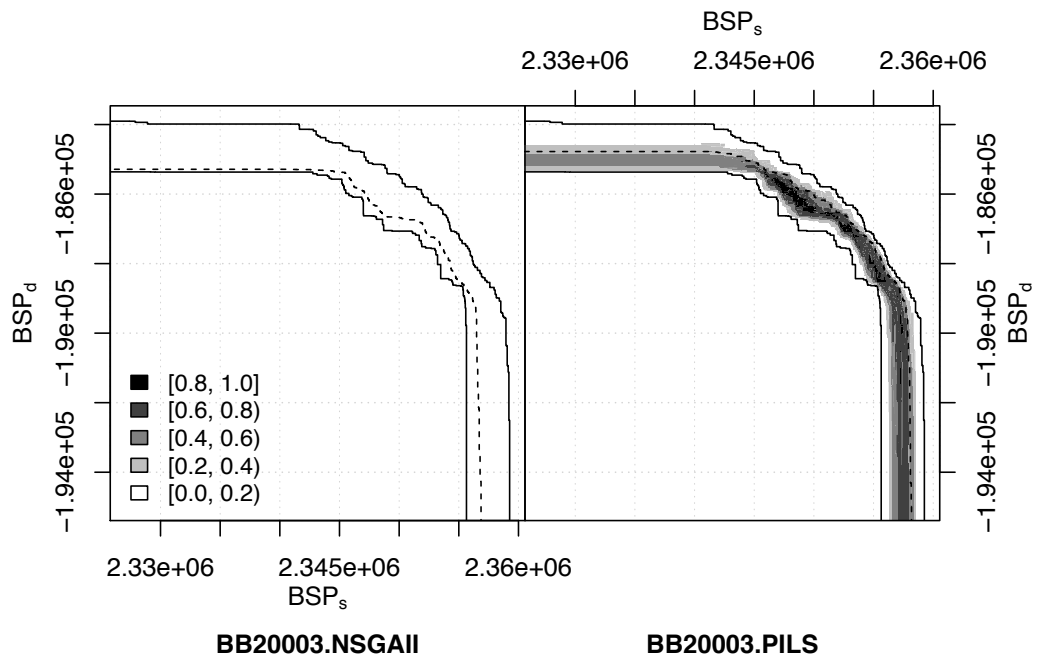


Figure A.23: Plot of the EAF for the instance BB20003

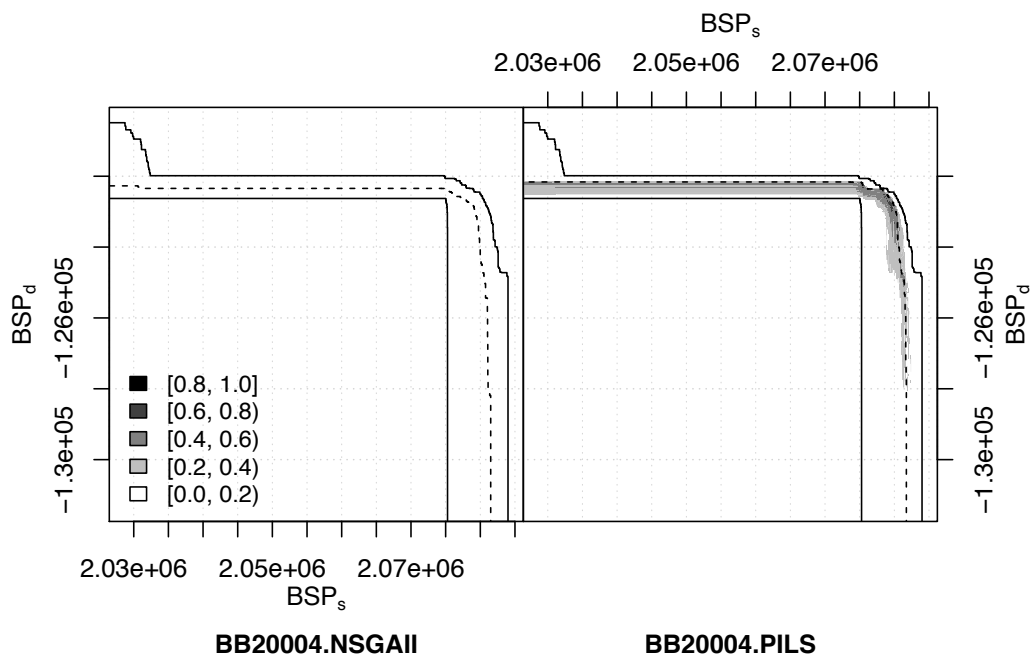


Figure A.24: Plot of the EAF for the instance BB20004

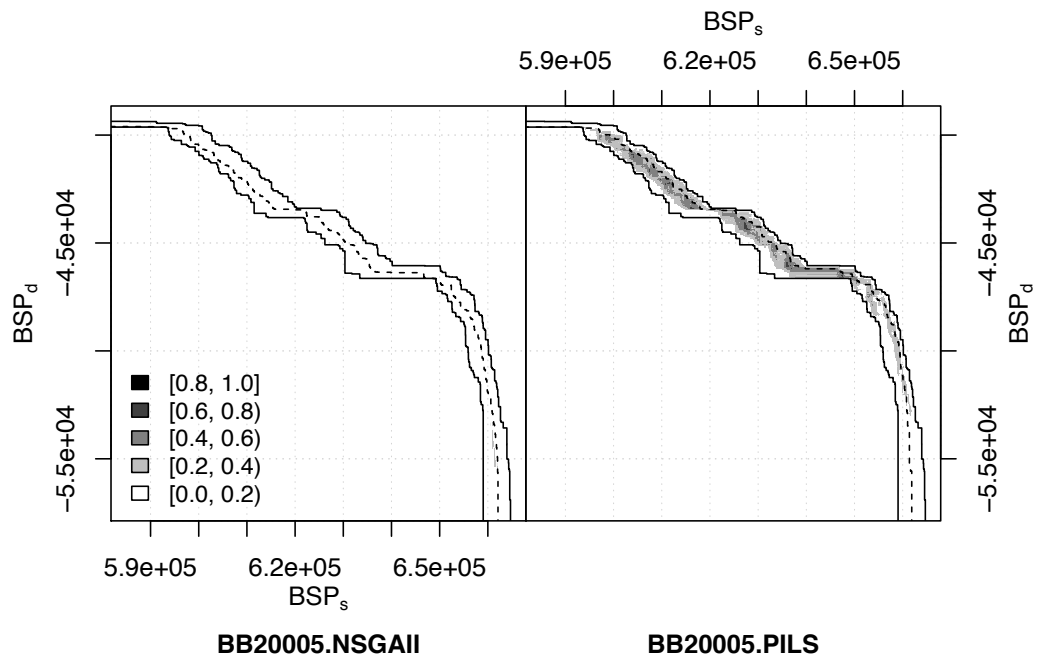


Figure A.25: Plot of the EAF for the instance BB20005

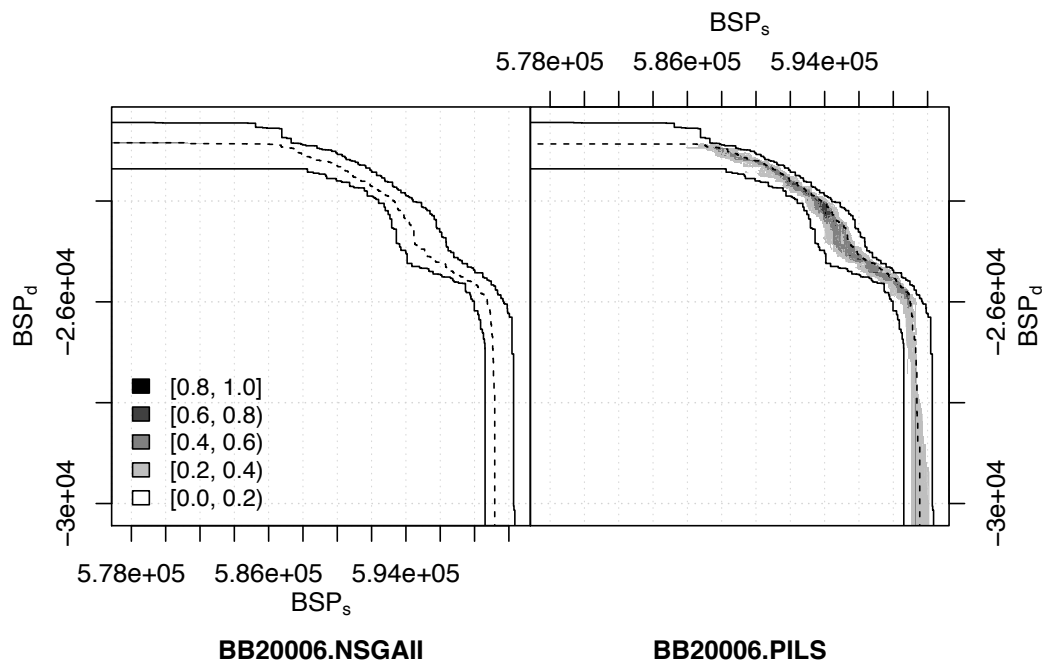


Figure A.26: Plot of the EAF for the instance BB20006

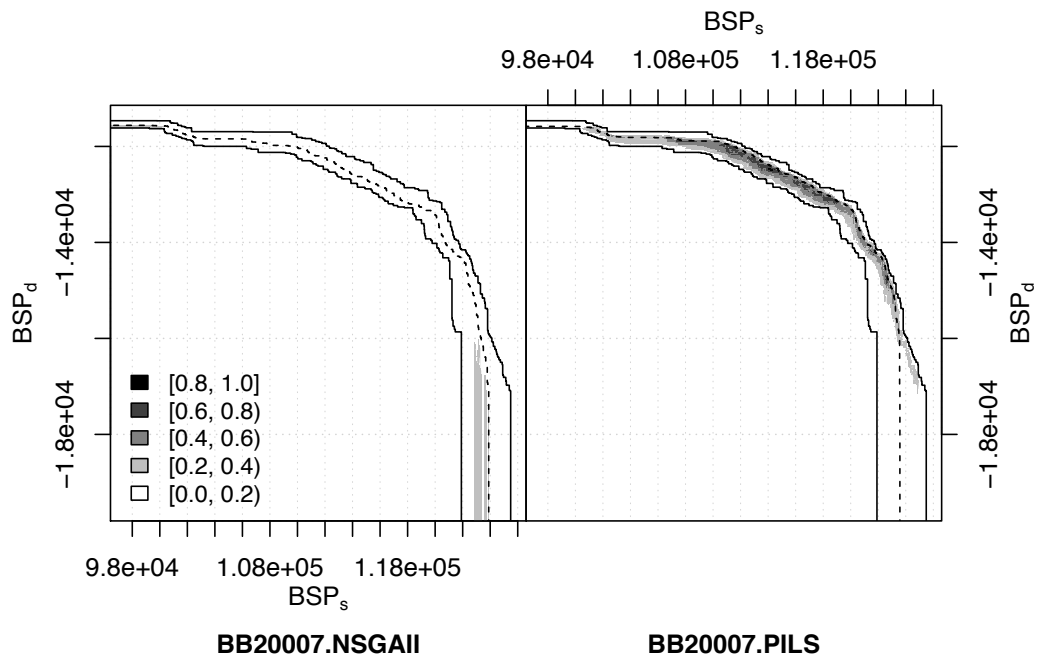


Figure A.27: Plot of the EAF for the instance BB20007

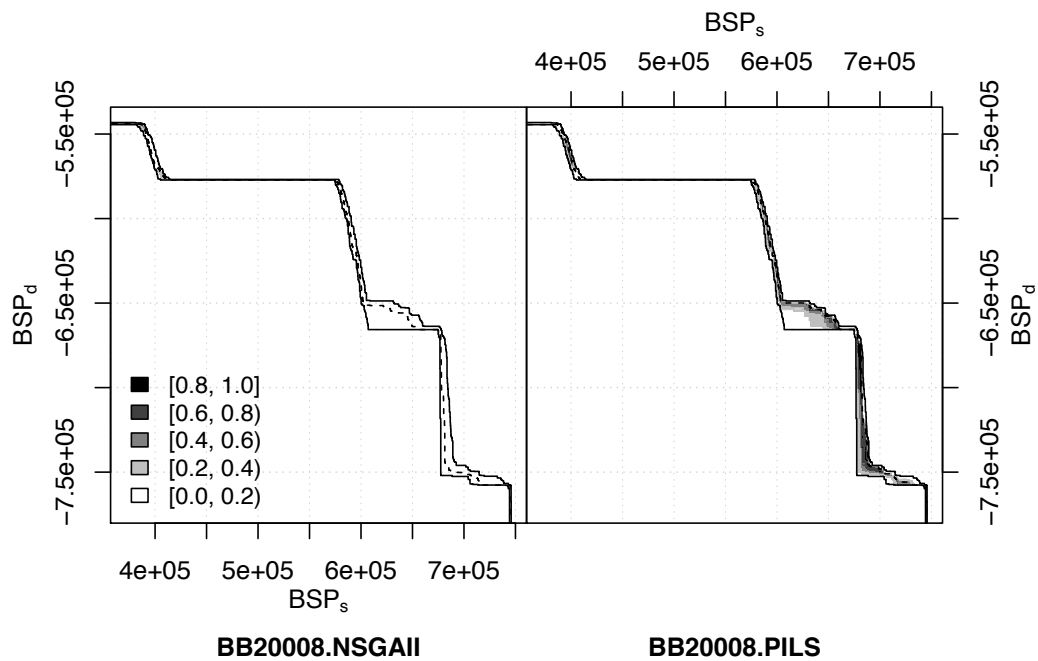


Figure A.28: Plot of the EAF for the instance BB20008

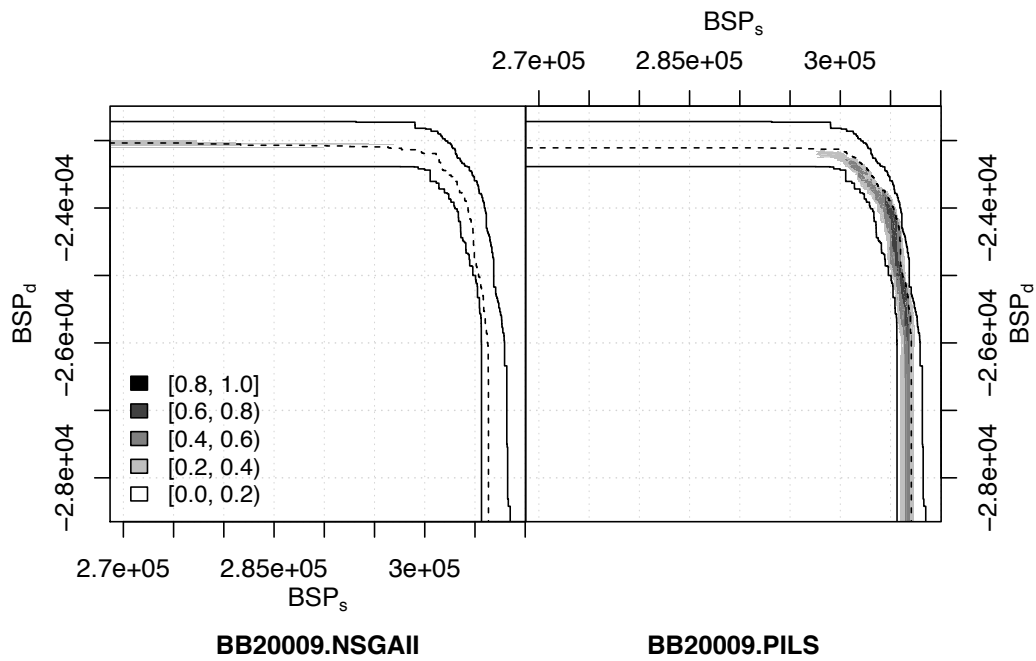


Figure A.29: Plot of the EAF for the instance BB20009

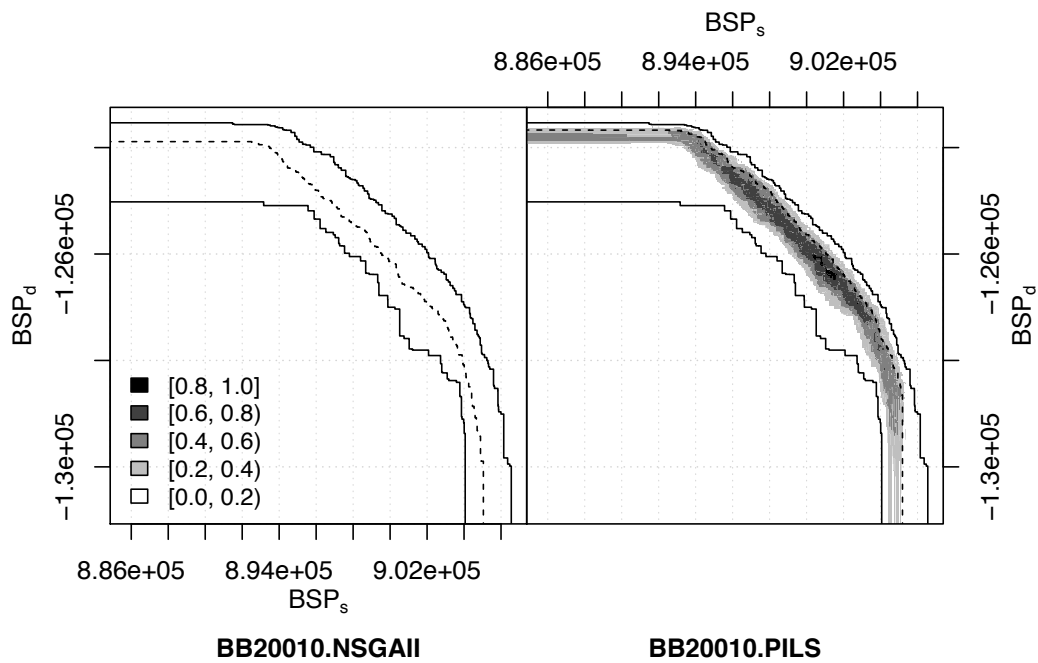


Figure A.30: Plot of the EAF for the instance BB20010

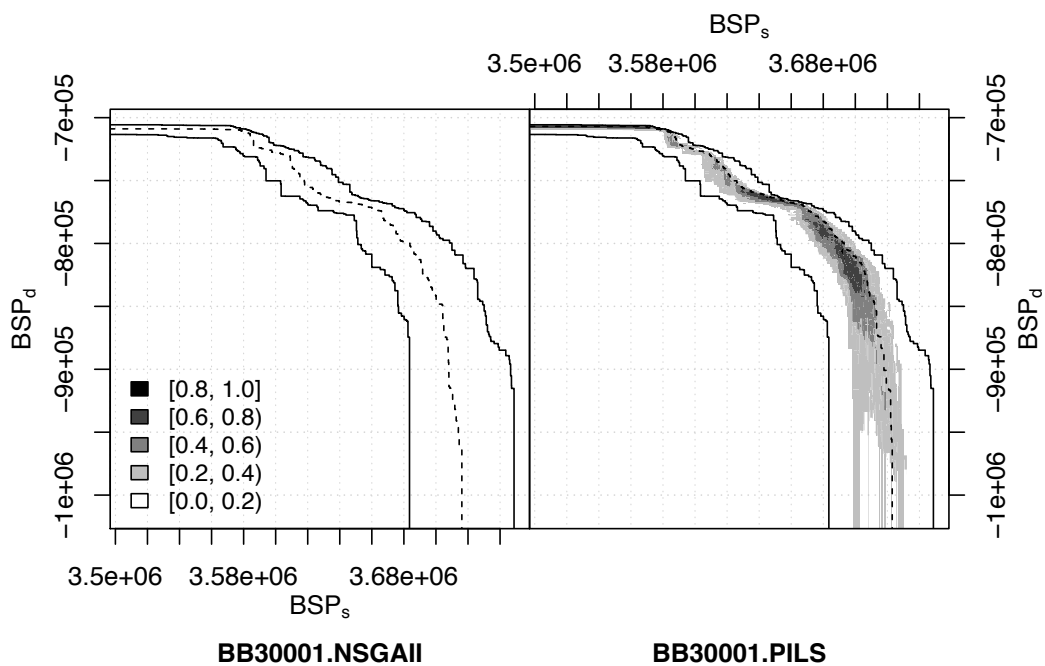


Figure A.31: Plot of the EAF for the instance BB30001

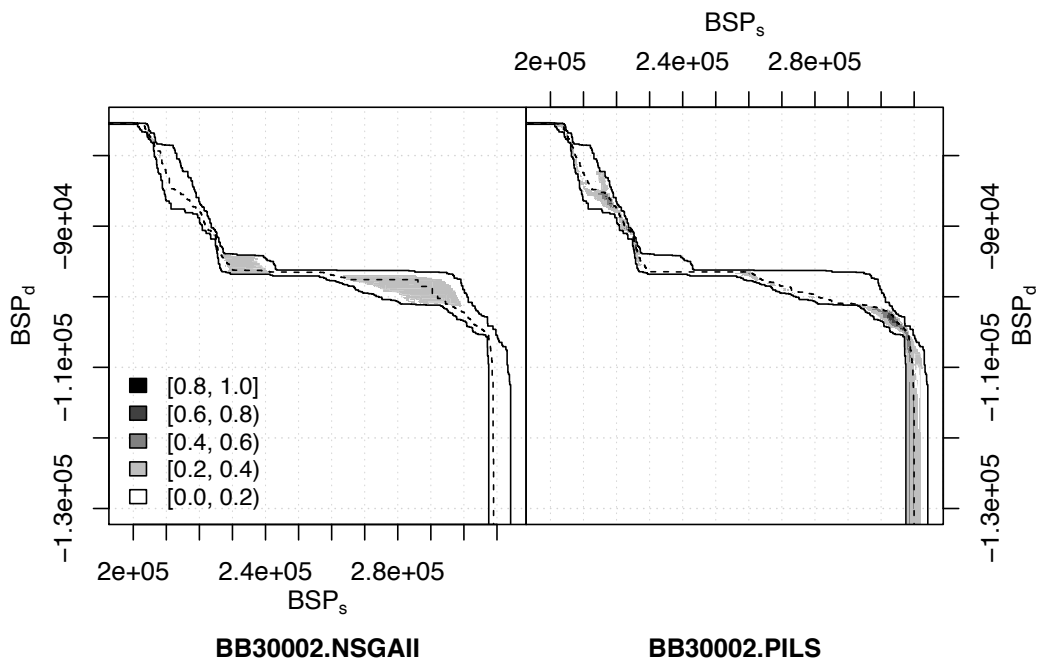


Figure A.32: Plot of the EAF for the instance BB30002

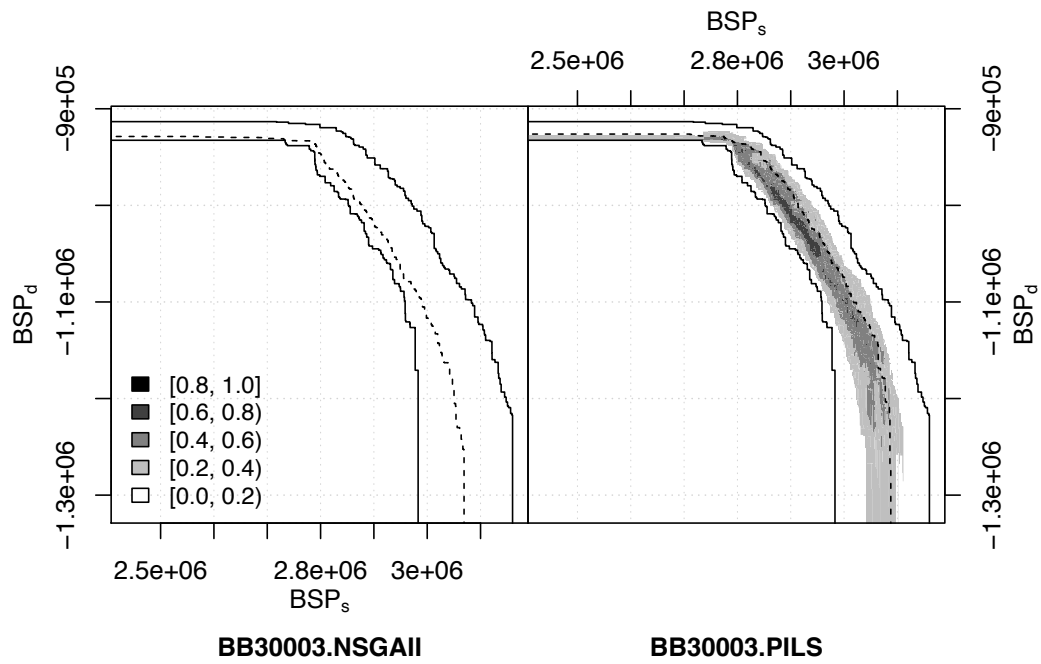


Figure A.33: Plot of the EAF for the instance BB30003

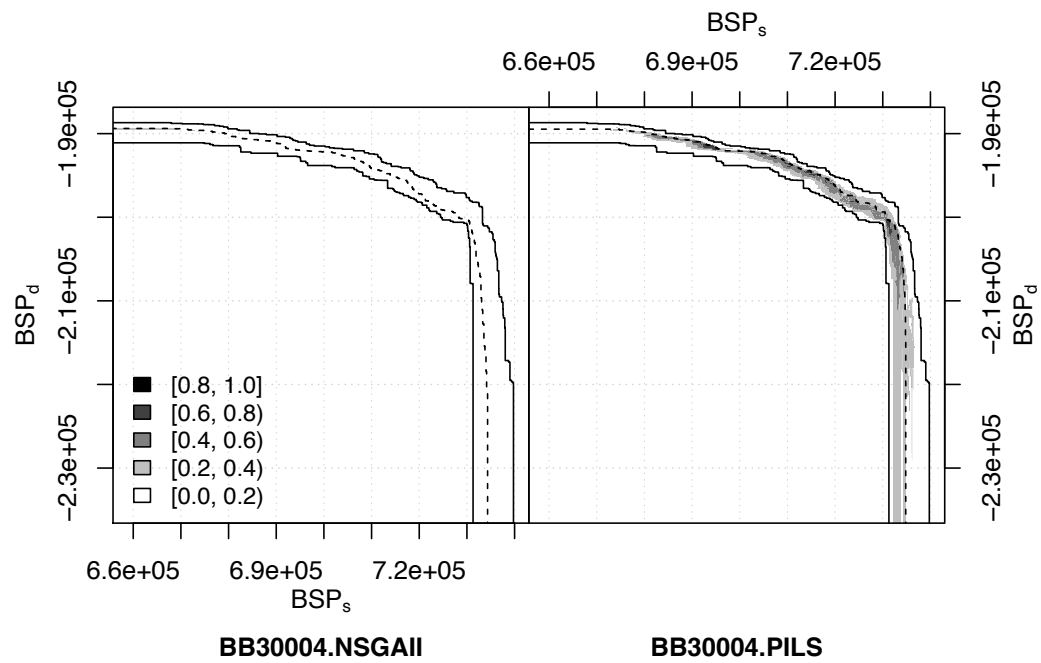


Figure A.34: Plot of the EAF for the instance BB30004

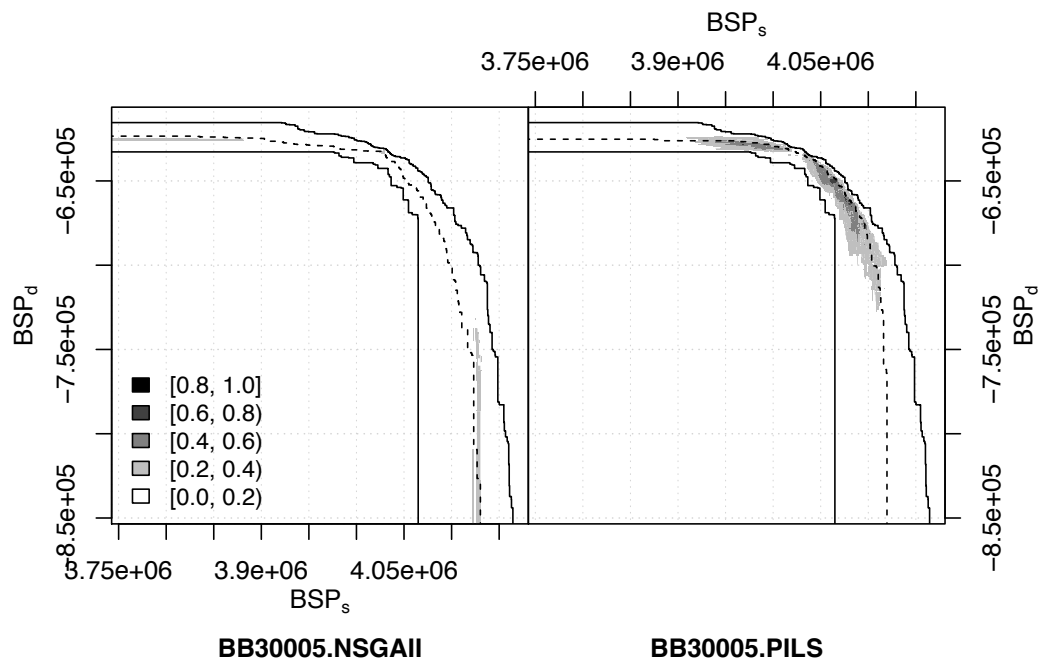


Figure A.35: Plot of the EAF for the instance BB30005

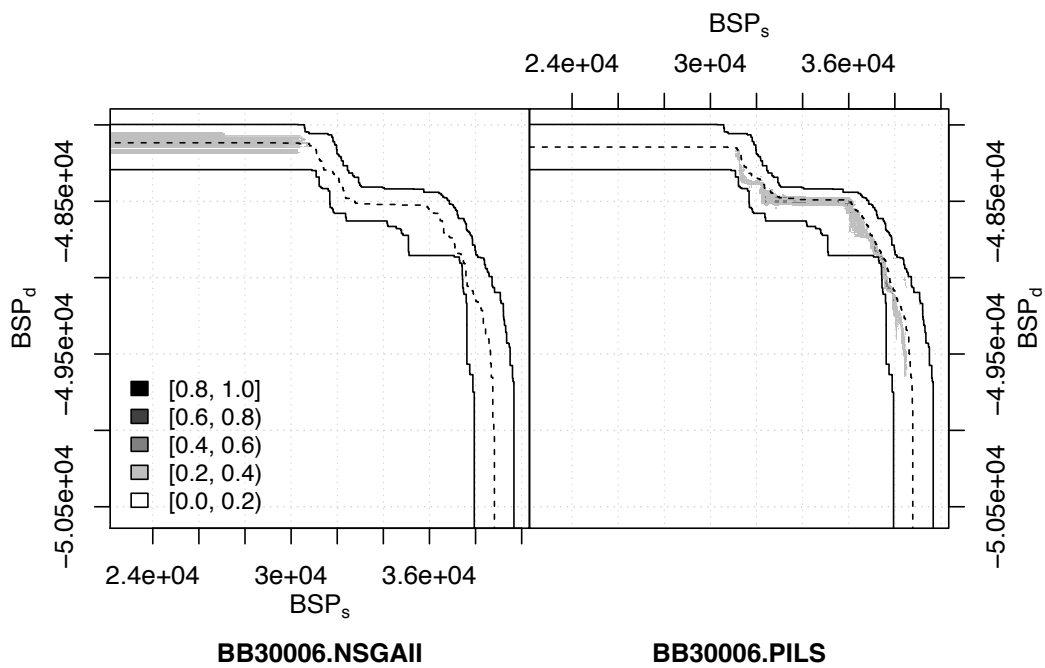


Figure A.36: Plot of the EAF for the instance BB30006

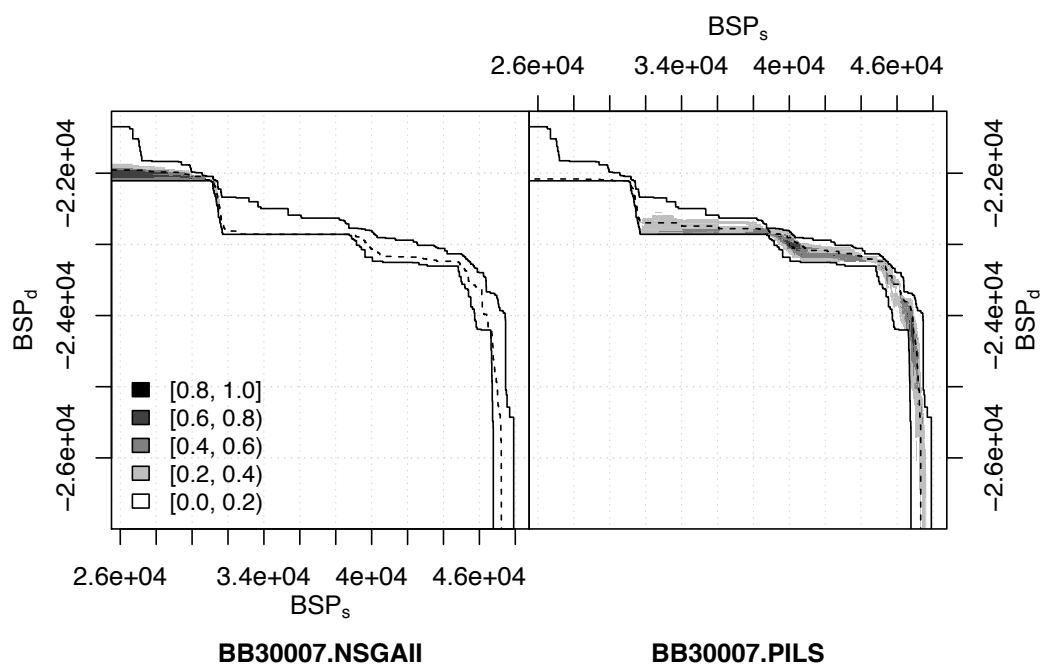


Figure A.37: Plot of the EAF for the instance BB30007

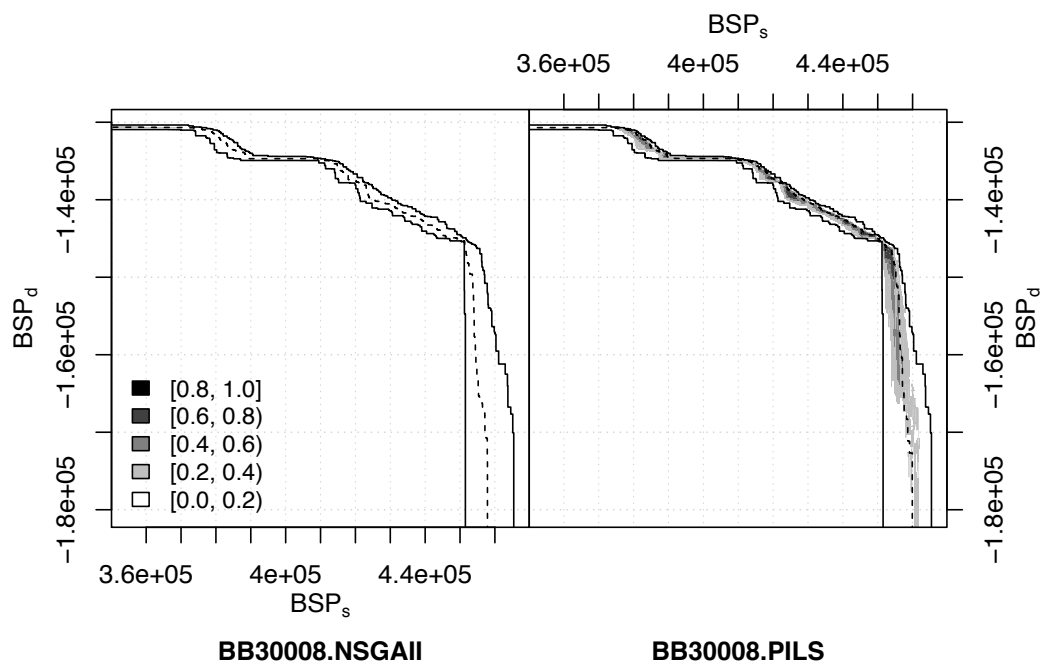


Figure A.38: Plot of the EAF for the instance BB30008

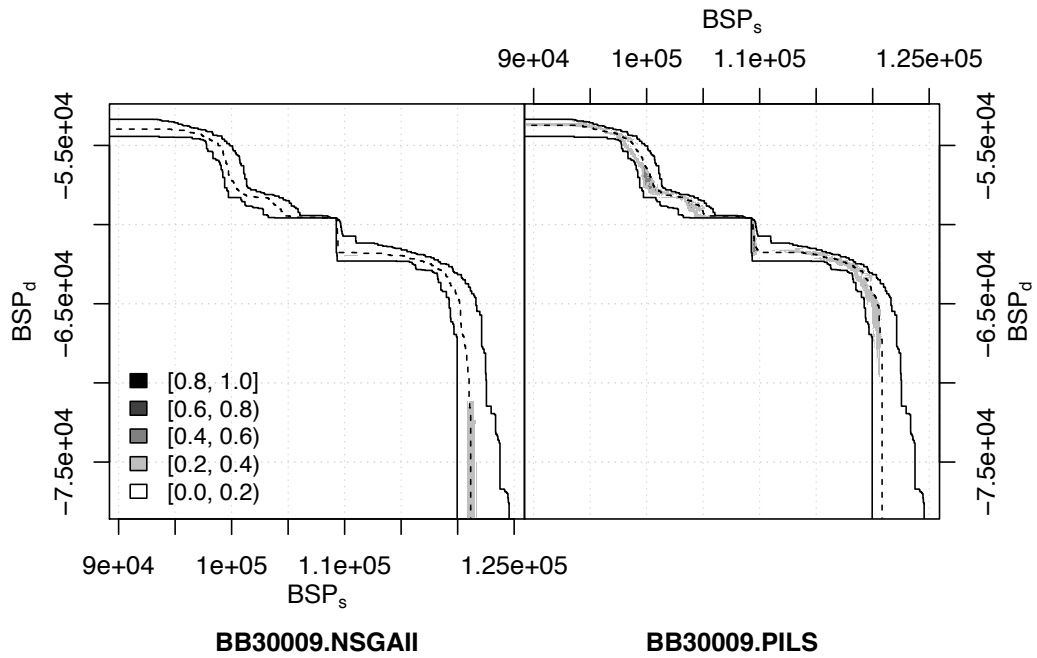


Figure A.39: Plot of the EAF for the instance BB30009

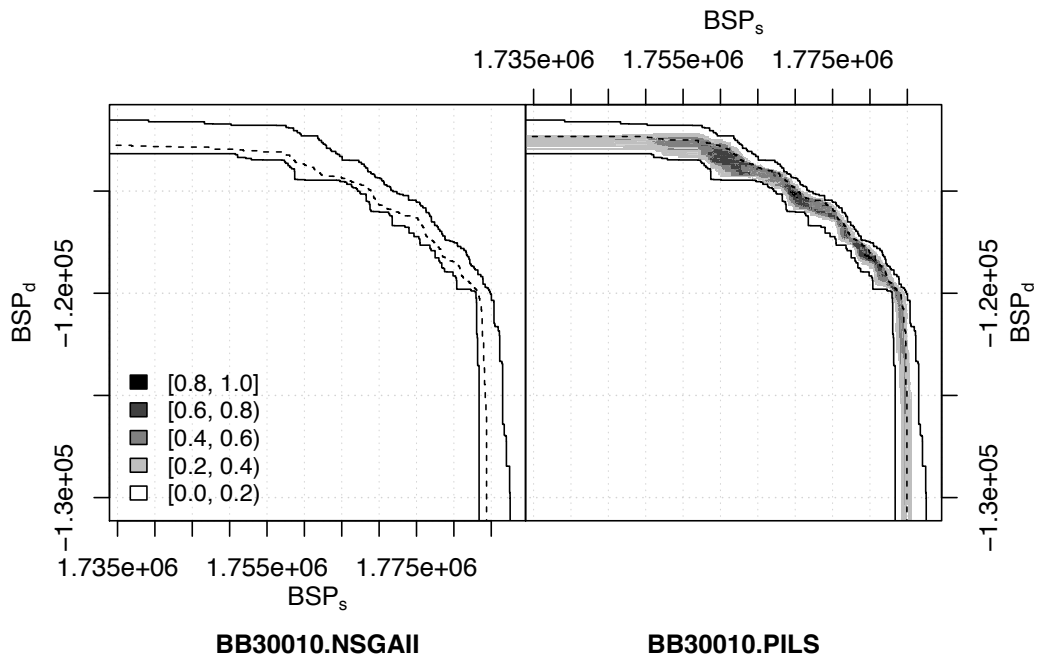


Figure A.40: Plot of the EAF for the instance BB30010

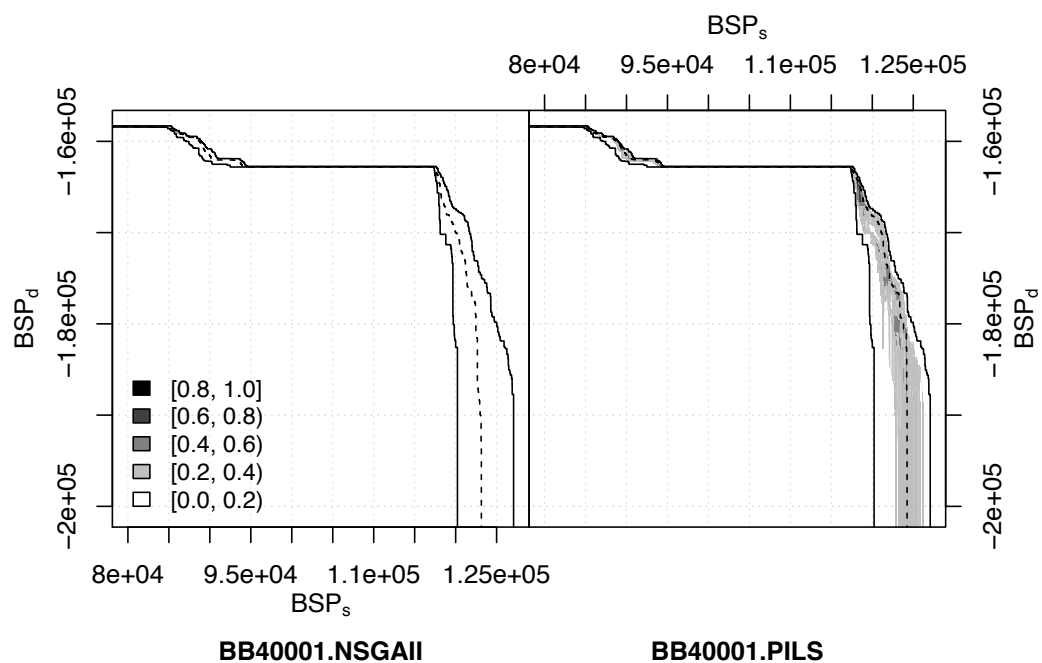


Figure A.41: Plot of the EAF for the instance BB40001

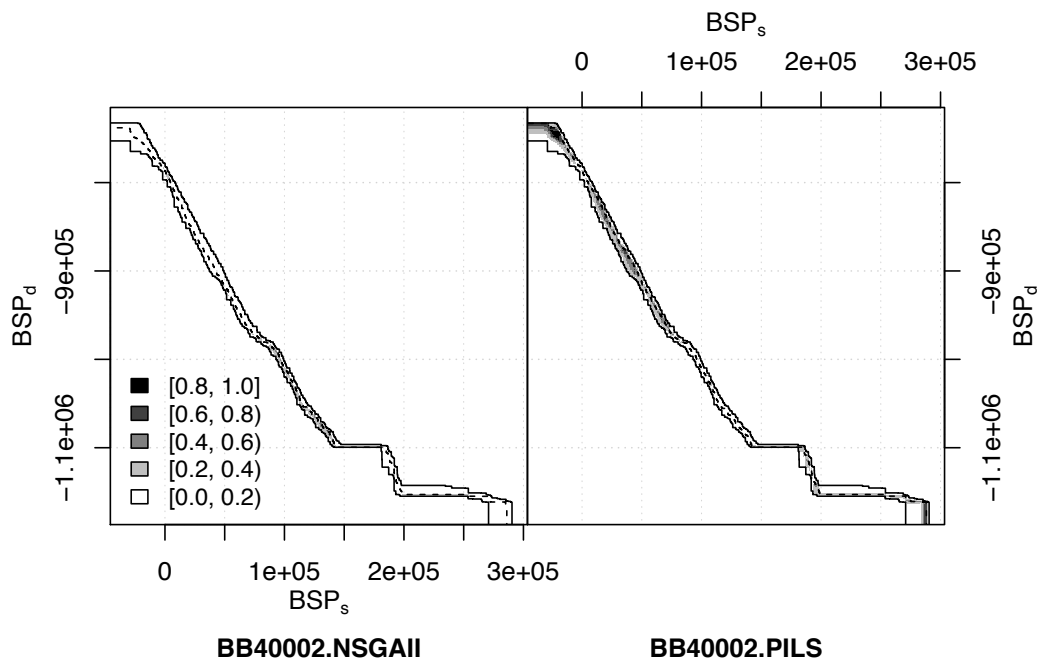


Figure A.42: Plot of the EAF for the instance BB40002

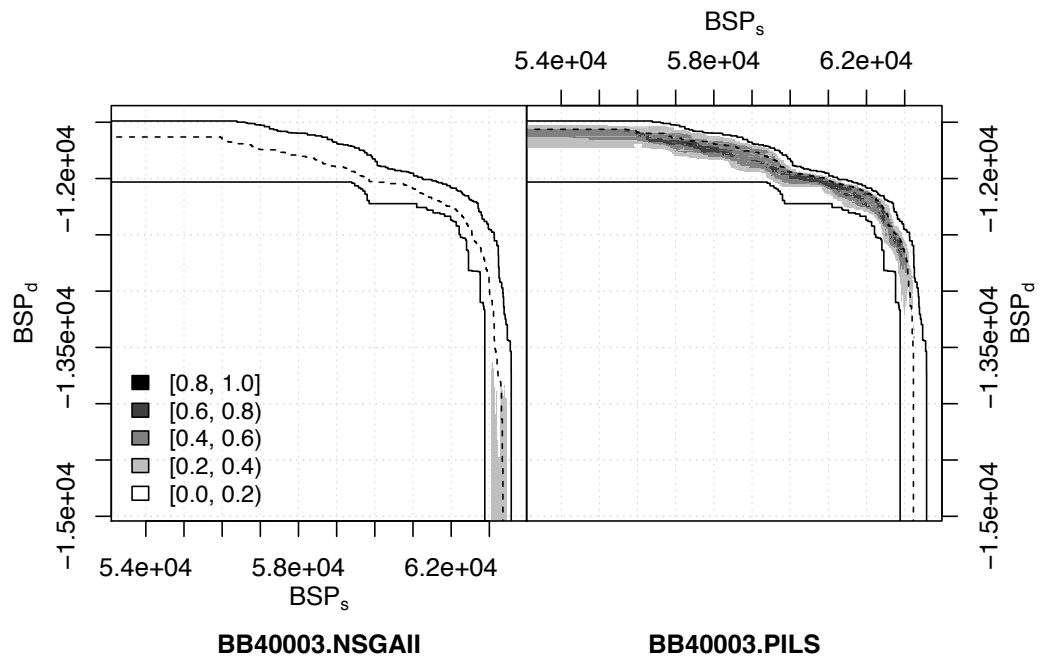


Figure A.43: Plot of the EAF for the instance BB40003

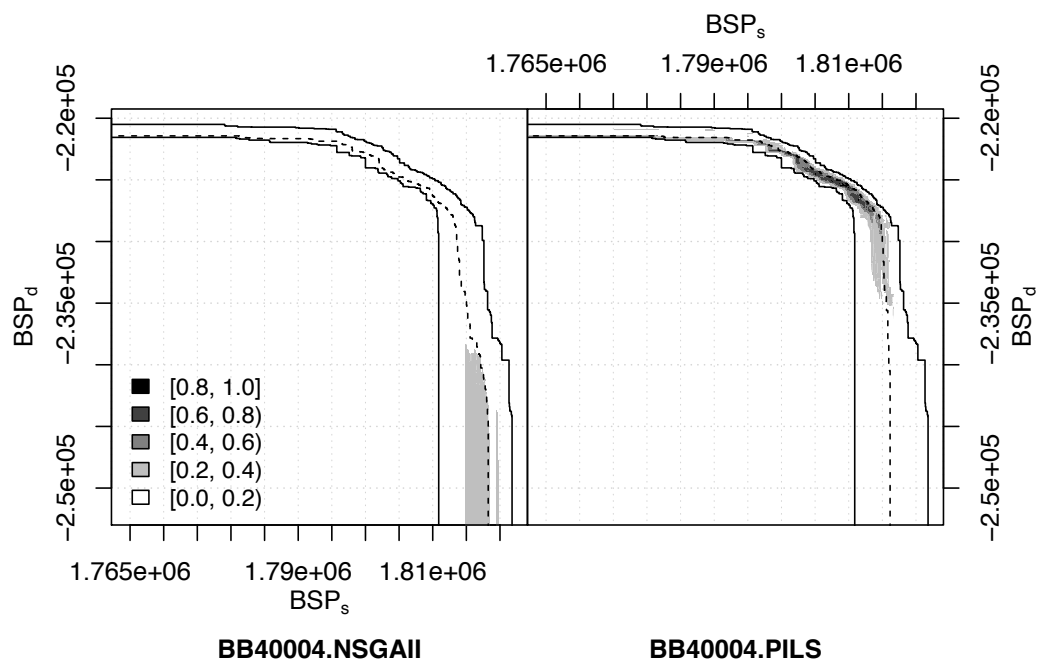


Figure A.44: Plot of the EAF for the instance BB40004

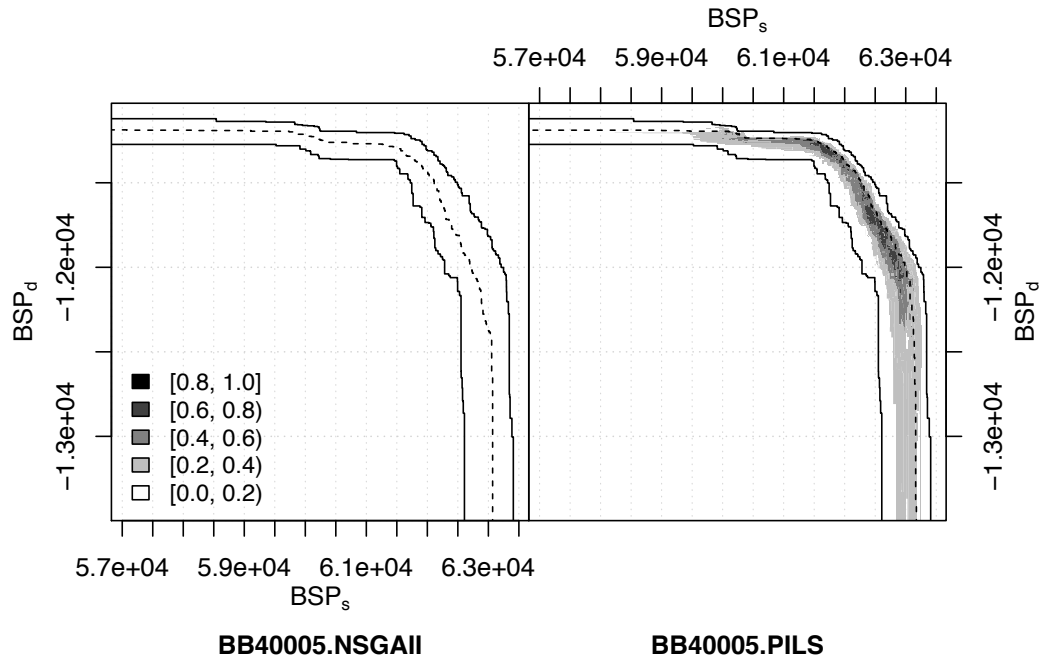


Figure A.45: Plot of the EAF for the instance BB40005

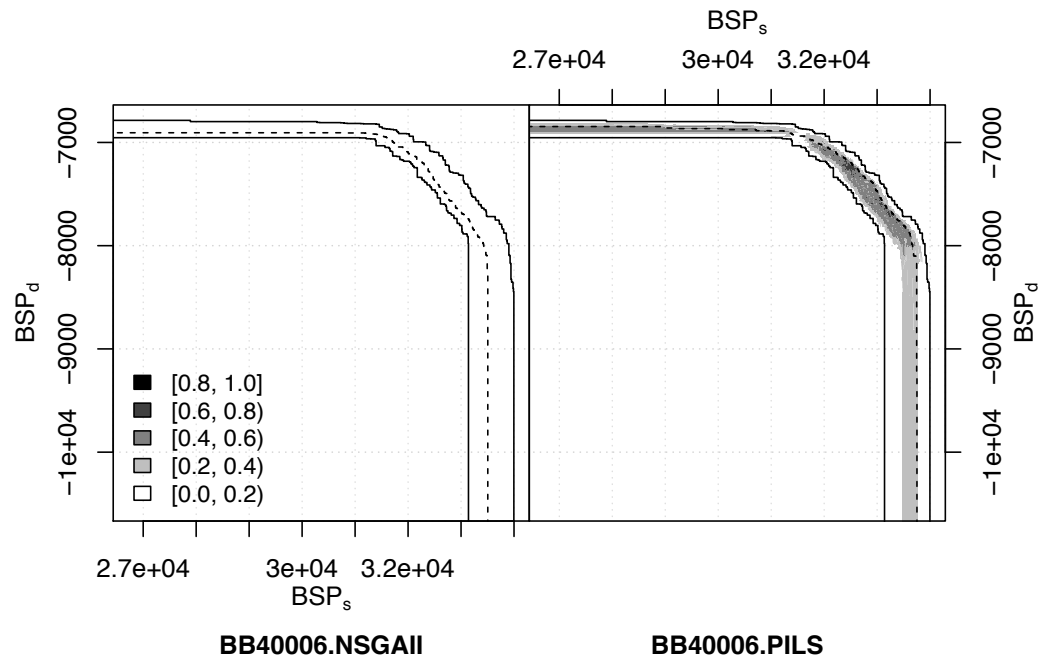


Figure A.46: Plot of the EAF for the instance BB40006

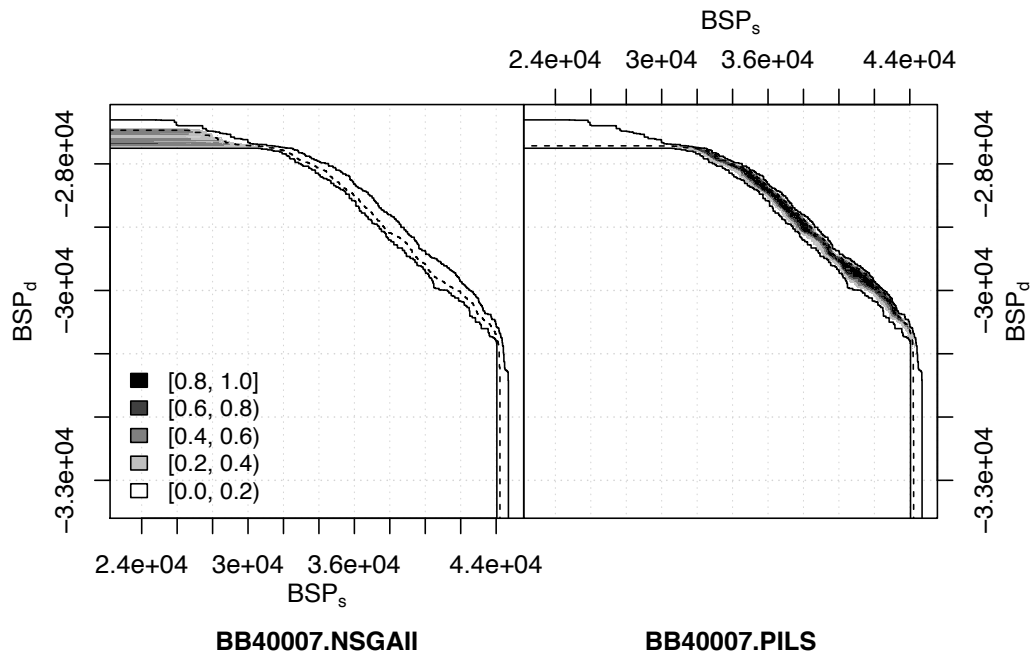


Figure A.47: Plot of the EAF for the instance BB40007

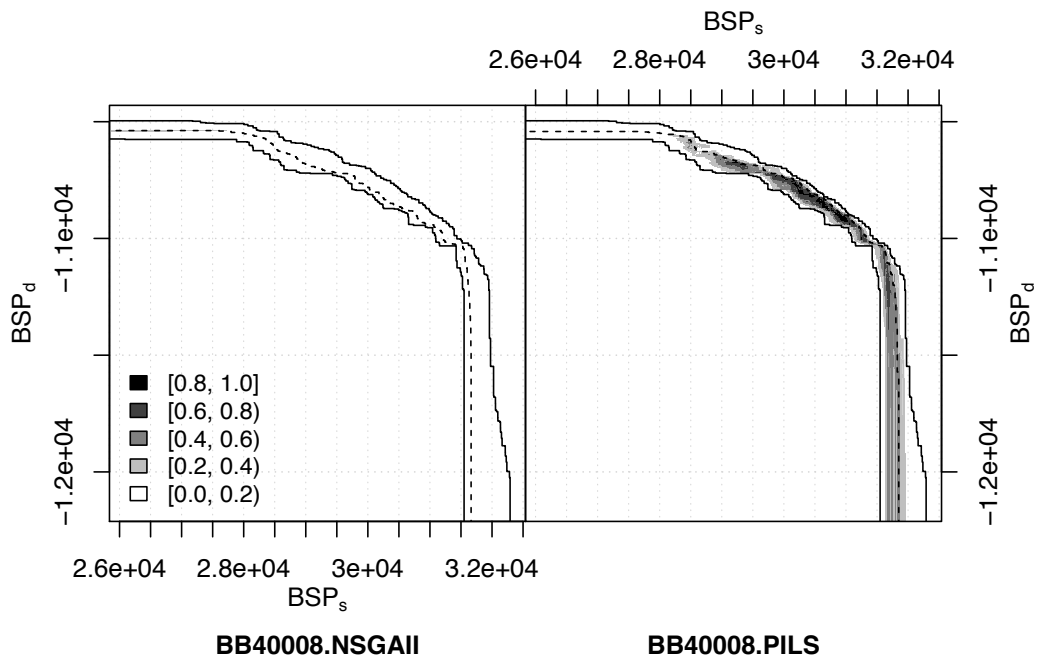


Figure A.48: Plot of the EAF for the instance BB40008

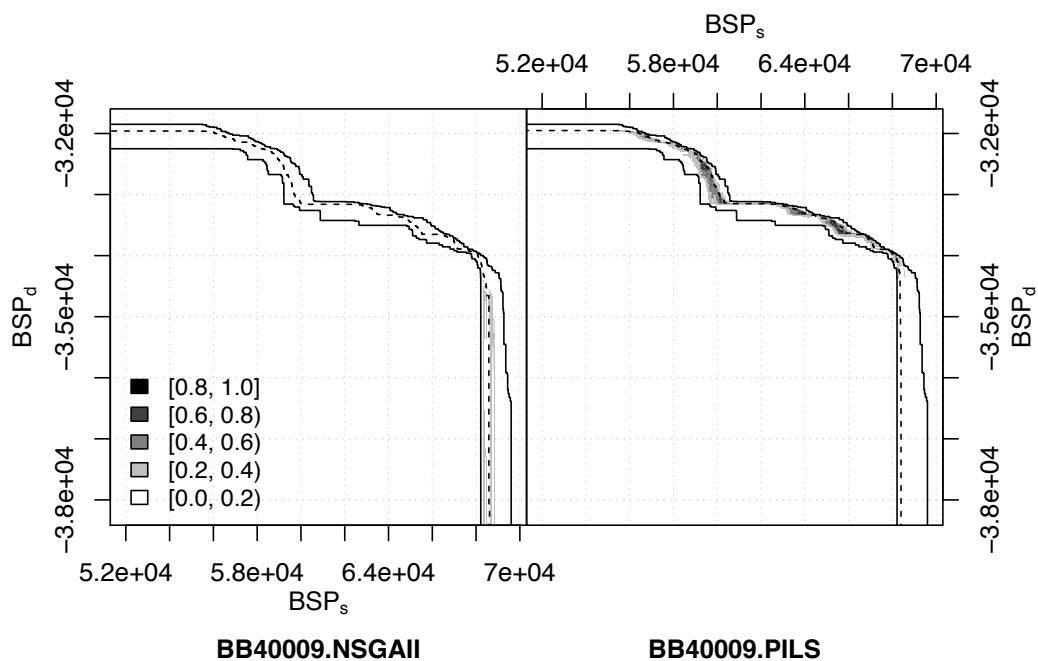


Figure A.49: Plot of the EAF for the instance BB40009

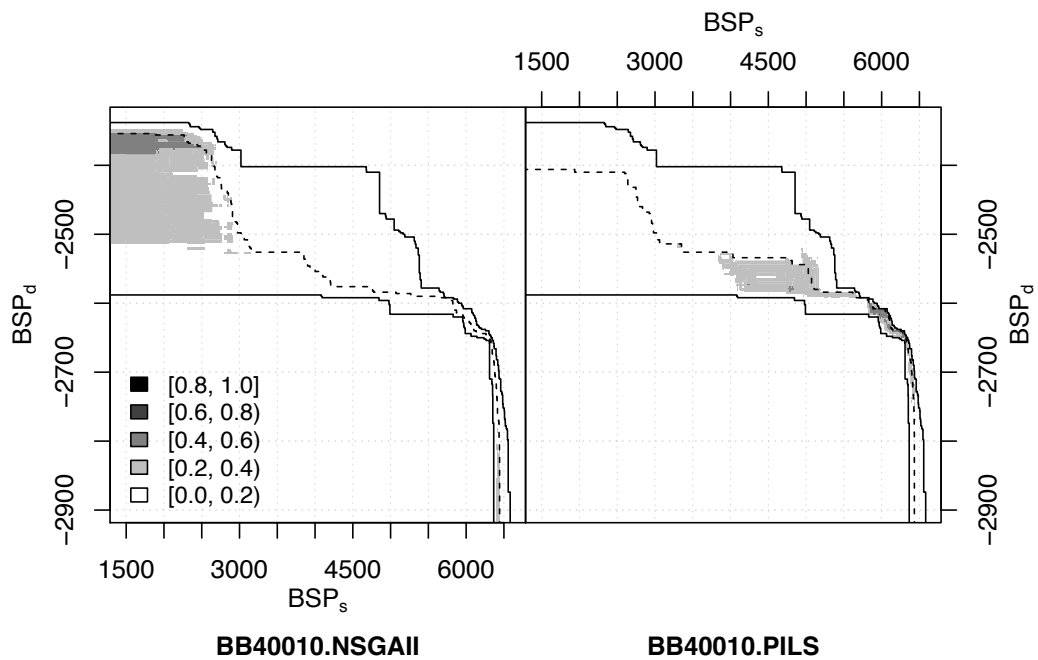


Figure A.50: Plot of the EAF for the instance BB40010

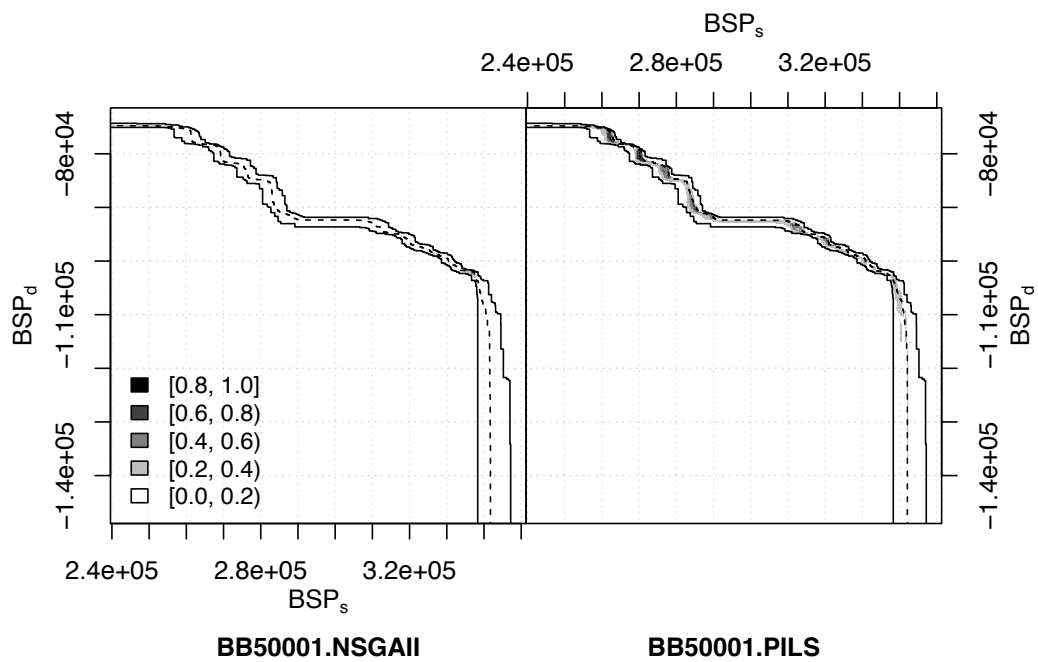


Figure A.51: Plot of the EAF for the instance BB50001

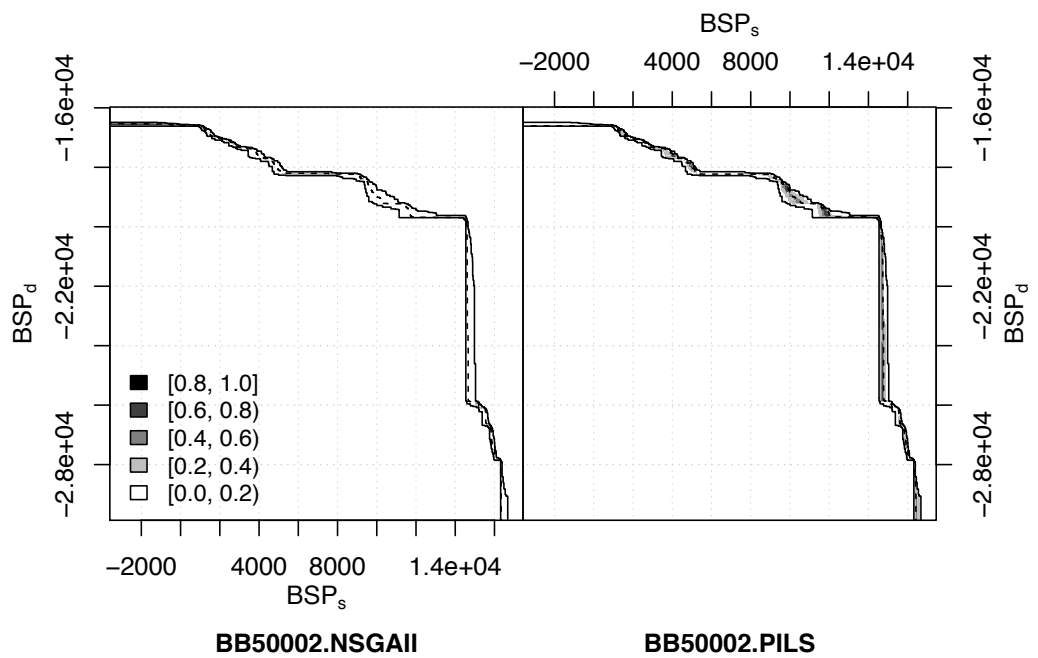


Figure A.52: Plot of the EAF for the instance BB50002

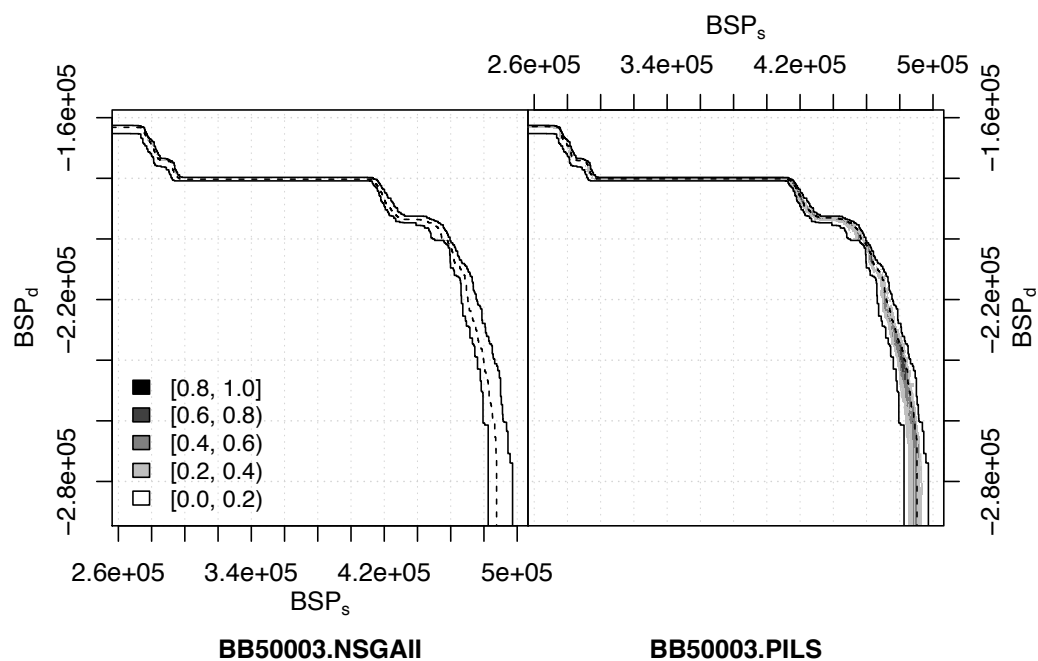


Figure A.53: Plot of the EAF for the instance BB50003

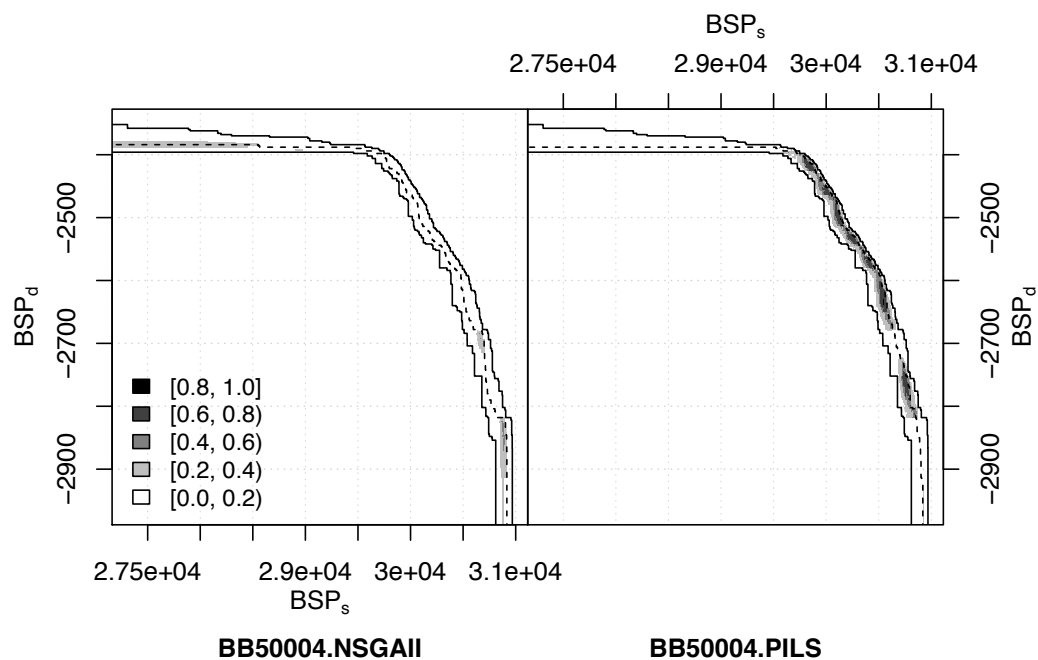


Figure A.54: Plot of the EAF for the instance BB50004

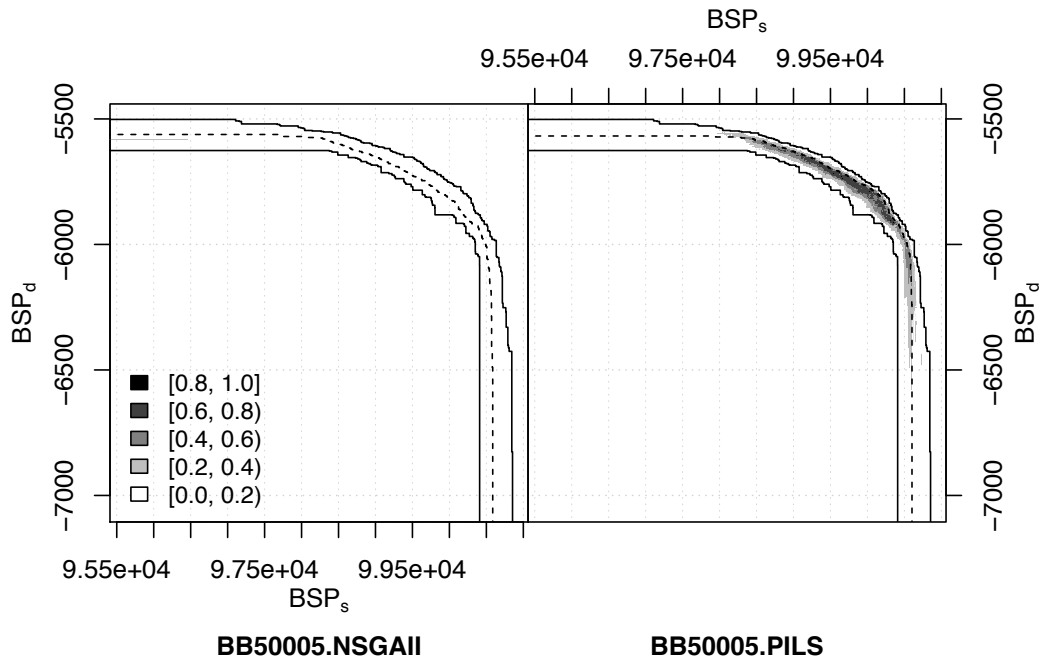


Figure A.55: Plot of the EAF for the instance BB50005

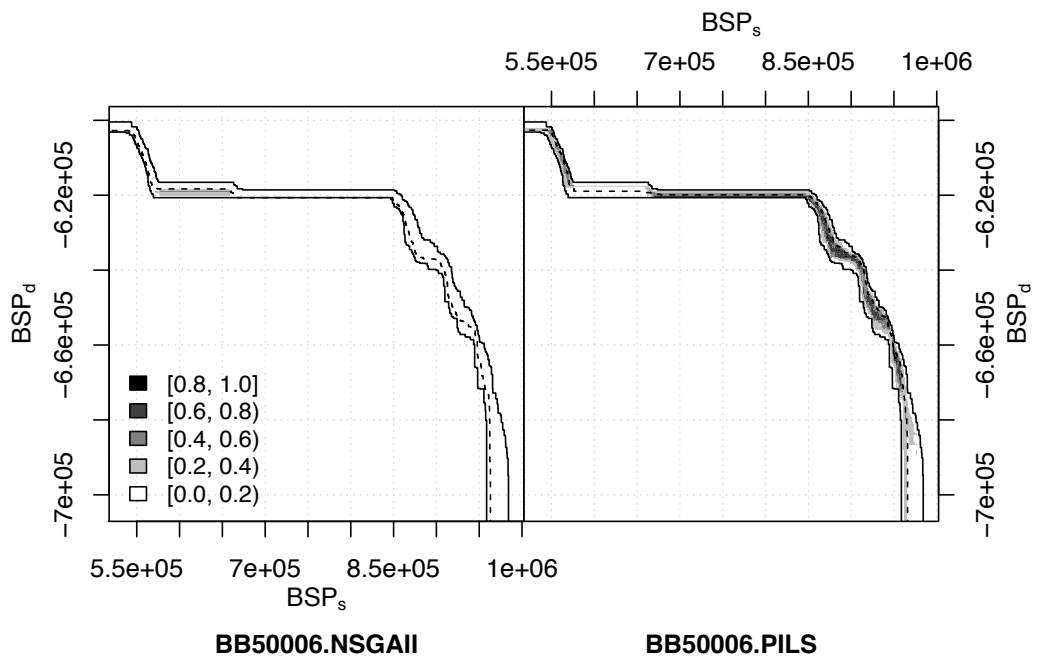


Figure A.56: Plot of the EAF for the instance BB50006

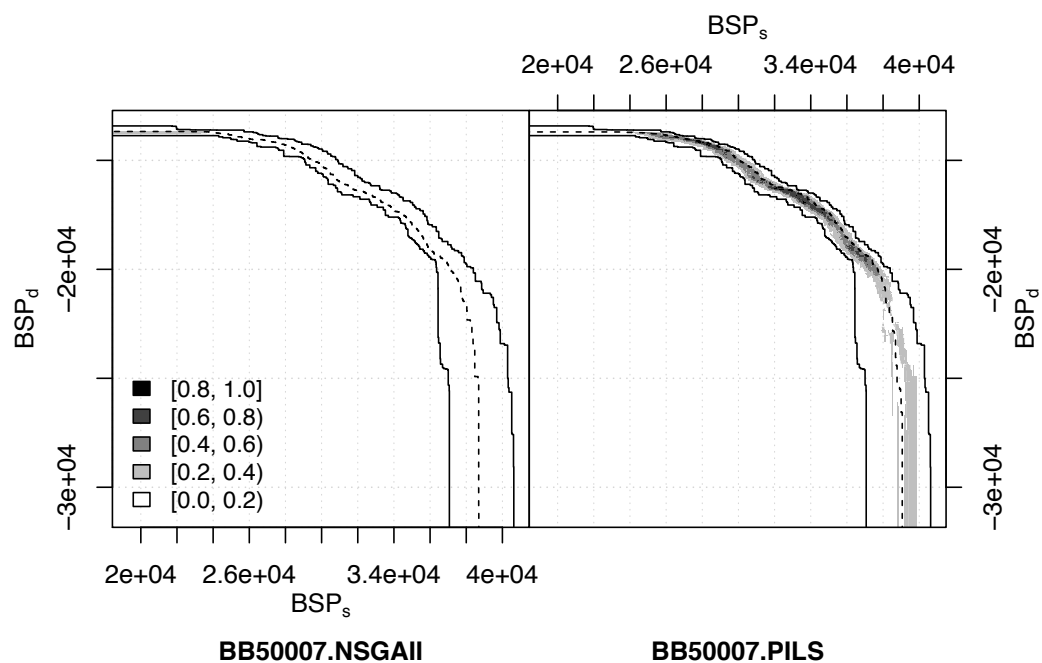


Figure A.57: Plot of the EAF for the instance BB50007

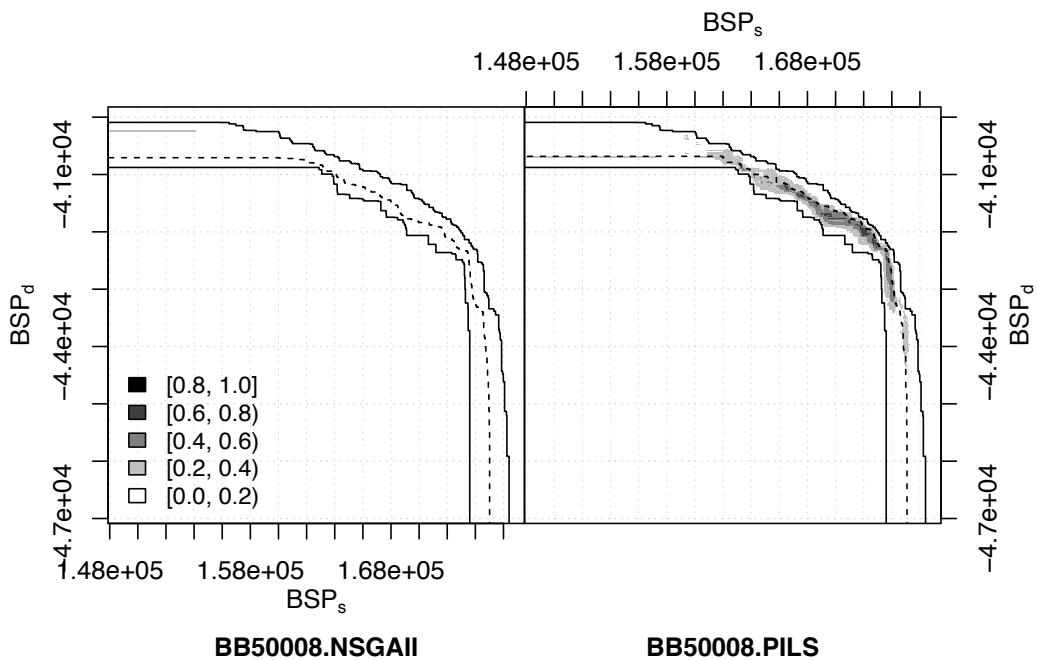


Figure A.58: Plot of the EAF for the instance BB50008

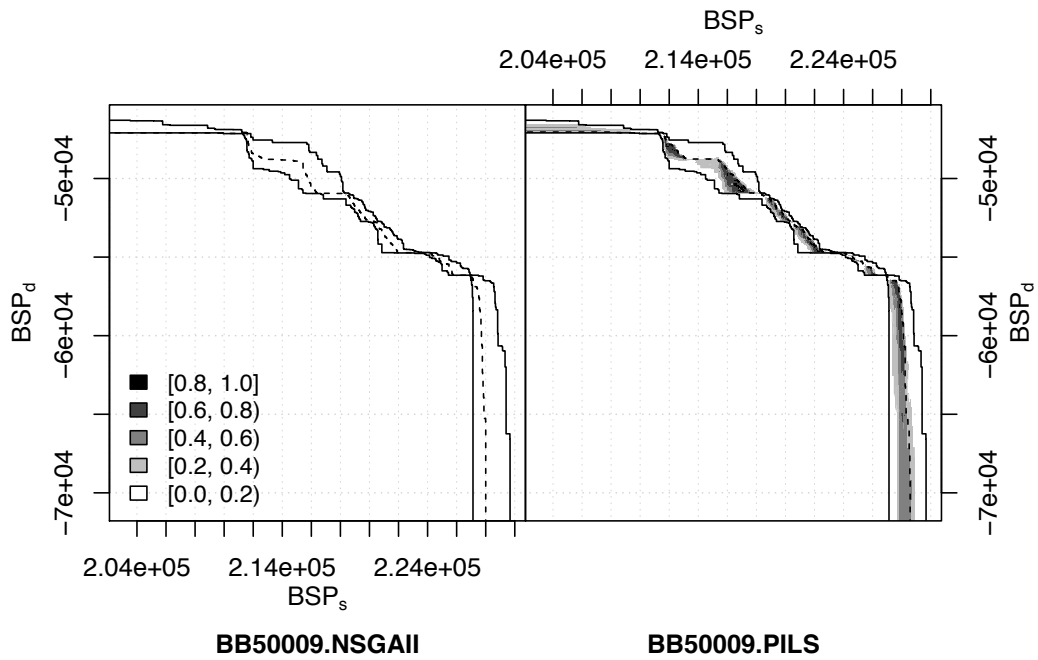


Figure A.59: Plot of the EAF for the instance BB50009

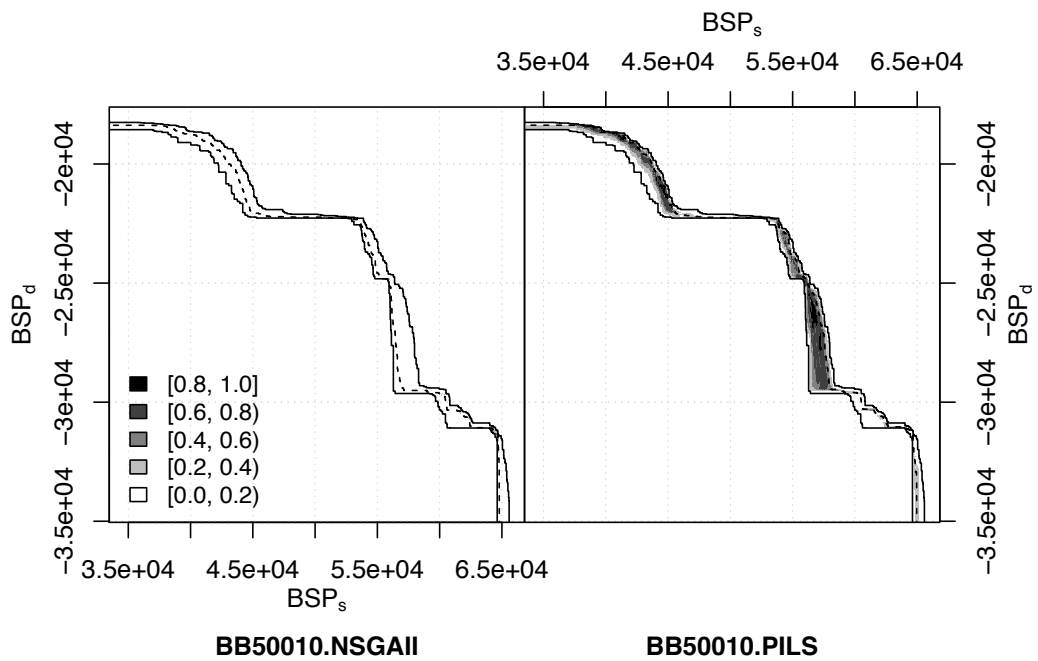


Figure A.60: Plot of the EAF for the instance BB50010

Bibliography

- Abbasi, M., Paquete, L., Liefoghe, A., and Dias, M. C. (2011). Multiobjective sequence alignment: Formulation and algorithms. 19th International Conference on Intelligent Systems for Molecular Biology and 10th European Conference on Computational Biology (ISMB/ECCB 2011), Vienna, Austria.
- Abbasi, M., Paquete, L., Liefoghe, A., Pinheiro, M., and Matias, P. (2013a). Improvements on bicriteria pairwise sequence alignment: algorithms and applications. *Bioinformatics*, 29(8):996–1003.
- Abbasi, M., Paquete, L., and Pereira, F. B. (2015). Local search for multiobjective multiple sequence alignment. In Ortuño, F. and Rojas, I., editors, *Third International Conference on Bioinformatics and Biomedical Engineering. IWBBIO 2015*, volume 9044 of *Lecture Notes in Computer Science*, pages 175–182. Springer.
- Abbasi, M., Paquete, L., and Pereira, F. B. (2016). Heuristics for multiobjective multiple sequence alignment. *Biomedical Engineering Online*, 15(1):70.
- Abbasi, M., Paquete, L., Pereira, F. B., and Schenker, S. (2013b). Local search for bicriteria multiple sequence alignment. page 40. German Conference on Bioinformatics (GCB2013), Göttingen.
- Abbasi, M., Paquete, L., and Pinheiro, M. (2012). Dynamic programming algorithms for biobjective sequence alignment. In Rocha, M. and Rocha, I., editors, *Bioinformatics Open Days*, Minho, Portugal.

- Barton, G. and Sternberg, M. (1987). A strategy for the rapid alignment of protein sequences: confidence levels from tertiary structure comparisons. *Journal of Molecular Biology*, 198:327–337.
- Beier, R. and Vöcking, B. (2011). The knapsack problem. In Vöcking, B., Alt, H., Dietzfelbinger, M., Reischuk, R., Scheideler, C., Vollmer, H., and Wagner, D., editors, *Algorithms Unplugged*, pages 375–381. Springer.
- Berger, M. and Munson, P. (1991). A novel randomized iterative strategy for aligning multiple protein sequences. *Computer Applications in Biosciences*, 7(4):479–484.
- Bishop, M. and Friday, A. E. (1985). Evolutionary trees from nucleic acid and protein sequences. *Proceedings of the Royal Society B: Biological Sciences*, 226(1244):271–302.
- Blackburne, B. P. and Whelan, S. (2012). Measuring the distance between multiple sequence alignments. *Bioinformatics*, 28(4):495–502.
- Bockenbauer, H. J. and Bongartz, D. (2007). *Algorithmic Aspects of Bioinformatics*. Springer Berlin Heidelberg.
- Bradley, R. K., Roberts, A., Smoot, M., Juvekar, S., Do, J., Dewey, C., Holmes, I., and Pachter, L. (2009). Fast statistical alignment. *PLOS Computational Biology*, 5(5):e1000392.
- Butler, G., Rasmussen, M. D., Lin, M. F., Santos, M. A., Sakthikumar, S., Munro, C. A., Rheinbay, E., Grabherr, M., Forche, A., Reedy, J. L., et al. (2009). Evolution of pathogenicity and sexual reproduction in eight candida genomes. *Nature*, 459(7247):657–662.
- Carrillo, H. and Lipman, D. (1988). The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48:1073–1082.
- Chellapilla, K. and Fogel, G. B. (1999). Multiple sequence alignment using evolutionary programming. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 1, pages 445–452. IEEE.
- Chor, B. and Tuller, T. (2006). Finding a maximum likelihood tree is hard. *Journal of the ACM*, 53(5):722–744.

- Coello, C. C., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer Science & Business Media.
- Cohen, J. (2004). Bioinformatics - an introduction for computer scientists. *ACM Computing Surveys*, 36:122–158.
- da Fonseca, V. G., Fonseca, C. M., and Hall, A. O. (2001). Inferential performance assessment of stochastic optimisers and the attainment function. In Zitzler, E., Thiele, L., Deb, K., Coello Coello, C. A., and Corne, D., editors, *Proceeding of the First International Conference on Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 213–225, Berlin, Heidelberg. Springer.
- da Silva, F. J. M., Sánchez Pérez, J. M., Gómez Pulido, J. A., and Vega Rodríguez, M. A. (2010). AlineaGA—a genetic algorithm with local search optimization for multiple sequence alignment. *Applied Intelligence*, 32(2):164–172.
- Dayhoff, M. O. and Schwartz, R. M. (1978). Chapter 22: A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*.
- de Bakker, P. I., Bateman, A., Burke, D. F., Miguel, R. N., Mizuguchi, K., Shi, J., Shirai, H., and Blundell, T. L. (2001). HOMSTRAD: adding sequence information to structure-based alignments of homologous protein families. *Bioinformatics*, 17(8):748–749.
- Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., and Schwefel, H.-P., editors, *Proceedings of the Sixth International Conference in Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 849–858. Springer.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- Do, C. B., Mahabhashyam, M. S., Brudno, M., and Batzoglou, S. (2005). ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15(2):330–340.
- Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis. Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

- Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797.
- Egan, A. N. and Crandall, K. A. (2006). Theory of phylogenetic estimation. *Evolutionary Genetics: Concepts and Case Studies*, 1:426–436.
- Egli, M. and Saenger, W. (2013). *Principles of Nucleic Acid Structure*. Springer Science & Business Media.
- Ehrgott, M. (2000). Hard to say it's easy — four reasons why combinatorial multiobjective programmes are hard. In Haimes, Y. Y. and Steuer, R. E., editors, *Proceedings of the XIVth International Conference on Multiple Criteria Decision Making (MCDM). Lecture Notes in Economics and Mathematical Systems*, volume 487, pages 69–80, Berlin, Heidelberg. Springer.
- Ehrgott, M. (2005). *Multicriteria Optimization*. Springer, Berlin, Heidelberg.
- Felsenstein, J. (1981). Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376.
- Felsenstein, J. (1985). Confidence Limits on Phylogenies: an Approach Using the Bootstrap. *Evolution*, 39:783–791.
- Feng, D.-F. and Doolittle, R. F. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360.
- Figueira, J. R., Fonseca, C. M., Halffmann, P., Klamroth, K., Paquete, L., Ruzika, S., Schulze, B., Stiglmayr, M., and Willems, D. (2017). Easy to say they are hard, but hard to see they are easy - towards a categorization of tractable multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 24(1-2):82–98.
- Fitch, W. M. (1971). Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416.
- Fonseca, C. M., Da Fonseca, V. G., and Paquete, L. (2005). Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In Coello Coello, C. A., Hernández Aguirre, A., and Zitzler, E., editors, *Proceeding of the Third International Conference on Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 250–264, Berlin, Heidelberg. Springer.

- Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo. Morgan Kaufmann.
- Fonseca, C. M. and Fleming, P. J. (1996). On the performance assessment and comparison of stochastic multiobjective optimizers. In Voigt, H.-M., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 584–593, Berlin, Heidelberg. Springer.
- Foulds, L. R. and Graham, R. L. (1982). The steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3(1):43–49.
- Gascuel, O. (1997). BIONJ: An Improved Version of the NJ Algorithm Based on a Simple Model of Sequence Data . *Molecular Biology and Evolution*, 14(7):685–695.
- Gascuel, O. and Steel, M. (2006). Neighbor-joining revealed. *Molecular Biology and Evolution*, 23(11):1997–2000.
- Goldberg, D. E., Richardson, J., et al. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. L. Erlbaum Associates Inc. Hillsdale, NJ, USA.
- Gotoh, O. (1996). Significant improvement in accuracy of multiple protein sequence alignments by iterative refinements as assessed by reference to structural alignments. *Journal of Molecular Biology*, 264(4):823–838.
- Gronau, I. and Moran, S. (2007). Optimal implementations of UPGMA and other common clustering algorithms. *Information Processing Letters*, 104(6):205–210.
- Gusfield, D. (1999). *Algorithms on Strings, Trees and Sequences* . Cambridge University Press.
- Gusfield, D., Balasubramanian, K., and Naor, D. (1994). Parametric optimization of sequence alignment. *Algorithmica*, 12(4–5):312–326.

- Haimes, Y., Lasdon, L., and Wismer, D. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):4–6.
- Handl, J., Kell, D. B., and Knowles, J. (2007). Multiobjective optimization in bioinformatics and computational biology. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):279–292.
- Hendy, M. D. and Penny, D. (1989). A framework for the quantitative study of evolutionary trees. *Systematic Zoology*, 38(4):297–309.
- Hoos, H. H. and Stützle, T. (2004). *Stochastic Local Search: Foundations and Applications*. Elsevier.
- Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. In Michalewicz, Z., editor, *Proceedings of the First IEEE Conference on Evolutionary Computation*, volume 1, pages 82–87. IEEE.
- Huang, X. and Miller, W. (1991). A time-efficient, linear-space local similarity algorithm. *Advances in Applied Mathematics*, 12(3):337–357.
- Huelsenbeck, J. P. (1995). Performance of phylogenetic methods in simulation. *Systematic Biology*, 44(1):17–48.
- Isaac, E. (2006). Settling the intractability of multiple alignment. *Journal of Computational Biology*, 13(7):1323–1339.
- Isaev, A. (2004). *Introduction to Mathematical Methods in Bioinformatics (Universitext)*, volume 6. Springer.
- Jain, H. and Deb, K. (2014). An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Transaction on Evolutionary Computation*, 18(4):602–622.
- Jensen, M. (2003). Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503–515.
- Jones, D. T. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202.

- Jones, N. and Pevzner, P. (2004). *An Introduction to Bioinformatics Algorithms*, volume 5. Cambridge University Press.
- Just, W. (2001). Computational complexity of multiple sequence alignment with sp-score. *Journal of Computational Biology*, 8(6):615–623.
- Katoh, K., Misawa, K., Kuma, K.-i., and Miyata, T. (2002). MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Research*, 30(14):3059–3066.
- Kaya, M., Sarhan, A., and Alhajj, R. (2014). Multiple sequence alignment with affine gap by using multi-objective genetic algorithm. *Computer Methods and Programs in Biomedicine*, 114(1):38–49.
- Kemena, C., Taly, J.-F., Kleinjung, J., and Notredame, C. (2011). STRIKE: evaluation of protein MSAs using a single 3D structure. *Bioinformatics*, 27(24):3385–3391.
- Kleinberg, J. and Tardos, E. (2006). *Algorithm Design*. Pearson Education.
- Knowles, J., Thiele, L., and Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers, Tech. Rep. *Computer Engineering and Networks Laboratory (TIK), ETH Zurich*, 214:327–332.
- Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172.
- Kuhn, T., Fonseca, C. M., Paquete, L., Ruzika, S., Duarte, M. M., and Figueira, J. R. (2016). Hypervolume subset selection in two dimensions: Formulations and algorithms. *Evolutionary Computation*, 24(3):411–425.
- Kumar, S., Stecher, G., and Tamura, K. (2016). MEGA7: molecular evolutionary genetics analysis version 7.0 for bigger datasets. *Molecular Biology and Evolution*, page msw054.
- Kung, H. T., Luccio, F., and Preparata, F. P. (1975). On finding the maxima of a set of vectors. *Journal of ACM*, 22(4):469–476.
- La, D., Sutch, B., and Livesay, D. R. (2005). Predicting protein functional sites with phylogenetic motifs. *Proteins: Structure, Function, and Bioinformatics*, 58(2):309–320.
- Lassmann, T. and Sonnhammer, E. L. (2005). Kalign an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics*, 6(1):298.

- Lipman, D. J., Altschul, S. F., and Kececioglu, J. D. (1989). A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences*, 86(12):4412–4415.
- López-Ibañez, M., Paquete, L., and Stützle, T. (2010). *Experimental Methods for the Analysis of Optimization Algorithms*, chapter Exploratory Analysis of Stochastic Local Search Algorithms in Biobjective Optimization, pages 209–222. Springer.
- Lourenço, H. R., Martin, O. C., and Stützle, T. (2013). Iterated Local Search: Handbook of Metaheuristics. volume 57 of *International Series in Operations Research & Management Science*, pages 321–353. Kluwer Academic Publishers.
- Lourenço, H. R., Martin, O. C., and Stützle, T. (2010). Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, pages 363–397. Springer.
- Michener, C. and Sokal, R. (1957). A quantitative approach to a problem in classification. *Evolution*, 11:130–162.
- Miller, B. L. and Shaw, M. J. (1996). Genetic algorithms with dynamic niche sharing for multimodal function optimization. In Fukuda, T. and Furuhashi, T., editors, *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, pages 786–791, Nagoya, Japan.
- Morgenstern, B., Frech, K., Dress, A., and Werner, T. (1998). DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics*, 14(3):290–294.
- Morrison, D. A. (2007). Increasing the efficiency of searches for the maximum likelihood tree in a phylogenetic analysis of up to 150 nucleotide sequences. *Systematic Biology*, 56(6):988–1010.
- Morrison, D. A. (2015). Is sequence alignment an art or a science? *Systematic Botany*, 40(1):14–26.
- Mote, J., Murthy, I., and Olson, D. L. (1991). A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53(1):81–92.
- Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., and Alba, E. (2009). Mocell: a cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746.

- Needleman, S. and Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453.
- Nelson, D. L., Lehninger, A. L., and Cox, M. M. (2008). *Lehninger Principles of Biochemistry*. Macmillan.
- Nixon, K. C. (1999). The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15(4):407–414.
- Notredame, C. (2002). Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144.
- Notredame, C., Higgins, D. G., and Heringa, J. (2000). T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–217.
- Notredame, C., Holm, L., and Higgins, D. G. (1998). COFFEE: an objective function for multiple sequence alignments. *Bioinformatics*, 14(5):407–422.
- Ortuño, F. M., Valenzuela, O., Rojas, F., Pomares, H., Florido, J. P., Urquiza, J. M., and Rojas, I. (2013). Optimizing multiple sequence alignments using a genetic algorithm based on three objectives: structural information, non-gaps percentage and totally conserved columns. *Bioinformatics*, page btt360.
- Pais, F. S.-M., de Ruy, P., Oliveira, G., and Coimbra, R. (2014). Assessing the efficiency of multiple sequence alignment programs. *Algorithms for Molecular Biology*, 9(4).
- Paquete, L., Abbasi, M., Pinheiro, M., and Matias, P. (2012a). Algorithms and applications of biobjective pairwise sequence alignment. The 2012 Mini EURO Conference on Computational Biology, Bioinformatics and Medicine (EURO-CBBM 2012).
- Paquete, L., Abbasi, M., Pinheiro, M., and Matias, P. (2012b). Algorithms for multiobjective sequence alignment. 2nd Workshop on Bio-Optimization.
- Paquete, L., Matias, P., Abbasi, M., and Pinheiro, M. (2014). MOSAL: Software tools for multiobjective sequence alignment. *Source Code for Biology and Medicine*, 9(2).
- Paquete, L., Schiavinotto, T., and Stützle, T. (2007). On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research*, 156(1):83–97.

- Paquete, L. and Stützle, T. (2007). Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In Gonzalez, T., editor, *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall / CRC.
- Pearson, W. R. and Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448.
- Przybylski, A., Gandibleux, X., and Ehrgott, M. (2010). A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3):371–386.
- Raghava, G., Searle, S. M., Audley, P. C., Barber, J. D., and Barton, G. J. (2003). OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinformatics*, 4(1):47.
- Randall T. Schuh, A. V. B. (2009). *Biological Systematics: Principles and Applications*. Cornell University Press, Ithaca, NY.
- Roytberg, M. A., Semionenkoy, M. N., and Tabolina, O. I. (1999). Pareto-optimal alignment of biological sequences. *Biophysics*, 44(4):565–577.
- Saborido, R., Ruiz, A. B., and Luque, M. (2017). Global WASF-GA: an evolutionary algorithm in multiobjective optimization to approximate the whole pareto optimal front. *Evolutionary Computation*, 25:309–349.
- Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425.
- Sankoff, D. (1972). Matching sequences under deletion/insertion constraint. *Proceedings of the National Academy of Sciences USA*, 69:4–6.
- Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In Grefenstette, J., editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.
- Shaw, K., Nortcliffe, A., Thompson, M., Love, J., Fleming, P., and Fonseca, C. (1999). Assessing the performance of multiobjective genetic algorithms for optimization of a batch process scheduling problem. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 1, pages 37–45. IEEE.

- Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Soding, J., Thompson, J. D., and Higgins, D. G. (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7:539.
- Sleator, R. D. (2011). Phylogenetics. *Archives of Microbiology*, 193(4):235–239.
- Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.
- Sneath, P. H., Sokal, R. R., et al. (1975). Numerical taxonomy. the principles and practice of numerical classification. *The Quarterly Review of Biology*, 50(4):525–526.
- Soto, W. and Becerra, D. (2014). A multi-objective evolutionary algorithm for improving multiple sequence alignments. In *Brazilian Symposium on Bioinformatics*, pages 73–82. Springer.
- Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248.
- Stamatakis, A. (2004). *Distributed and parallel algorithms and systems for inference of huge phylogenetic trees based on the maximum likelihood method*. PhD thesis, Technical University Munich.
- Stothard, P. (2000). The sequence manipulation suite: Javascript programs for analyzing and formatting protein and DNA sequences. *Biotechniques*, 28(6):1102–1104.
- Szabó, A., Novák, Á., Miklós, I., and Hein, J. (2010). Reticular alignment: a progressive corner-cutting method for multiple sequence alignment. *BMC bioinformatics*, 11(1):570.
- Tamura, K., Peterson, D., Peterson, N., Stecher, G., Nei, M., and Kumar, S. (2011). MEGA5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. *Molecular Biology and Evolution*, 28(10):2731–2739.
- Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680.

- Thompson, J. D., Koehl, P., Ripp, R., and Poch, O. (2005). BALiBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins: Structure, Function, and Bioinformatics*, 61(1):127–136.
- Thompson, J. D., Plewniak, F., and Poch, O. (1999). A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27(13):2682–2690.
- Vaz, D., Paquete, L., Fonseca, C., Klamroth, K., and Stiglmayr, M. (2015). Representation of the non-dominated set in biobjective discrete optimization. *Computers & Operations Research*, 63:172–186.
- Warnow, T. (2012). Standard maximum likelihood analyses of alignments with gaps can be statistically inconsistent. *PLoS currents*, 4.
- Waterman, M. (1994). Parametric and ensemble sequence alignment. *Bulletin of Mathematical Biology*, 56(4):743–767.
- Wong, K. M., Suchard, M. A., and Huelsenbeck, J. P. (2008). Alignment uncertainty and genomic analysis. *Science*, 319(5862):473–476.
- Zambrano-Vega, C., Nebro, A. J., García-Nieto, J., and Aldana-Montes, J. F. (2017). Comparing multi-objective metaheuristics for solving a three-objective formulation of multiple sequence alignment. *Progress in Artificial Intelligence*, pages 1–16.
- Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.
- Zhu, H., He, Z., and Jia, Y. (2016). A novel approach to multiple sequence alignment using multiobjective evolutionary algorithm based on decomposition. *IEEE Journal of Biomedical and Health Informatics*, 20(2):717–727.
- Zitzler, E. and Thiele, L. (1998). Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature - PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–304, London, UK. Springer.
- Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

-
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.