A note on the ϵ-indicator subset selection

Daniel Vaz, Luís Paquete, Aníbal Ponte

PII: S0304-3975(13)00364-2
DOI: 10.1016/j.tcs.2013.05.013
Reference: TCS 9344

To appear in: *Theoretical Computer Science*

Received date: 9 February 2013
Revised date: 18 April 2013
Accepted date: 13 May 2013

Please cite this article in press as: D. Vaz et al., A note on the ϵ-indicator subset selection, *Theoretical Computer Science* (2013), http://dx.doi.org/10.1016/j.tcs.2013.05.013

A note on the ϵ-indicator subset selection

Daniel Vaz[a], Luís Paquete[a,*], Aníbal Ponte[b]

[a] *CISUC, Department of Informatics Engineering, University of Coimbra, Portugal*
[b] *ESTSETÚBAL, Polytechnic Institute of Setúbal, Portugal*

Abstract

The ϵ-indicator subset selection selects a subset of a nondominated point set that is as close as possible to a reference point set with respect to the ϵ-indicator. This selection procedure is used by population-based heuristic approaches for multiobjective optimization problems. Given that this procedure is called very often during the run of the heuristic approach, efficient ways of computing the optimal subset are strongly required. In this note, we give a correctness proof of the ϵ-indicator subset selection algorithm proposed by Ponte et al. [1] for the bidimensional case as well as several algorithmic improvements in terms of time complexity. Extensions to larger dimension are also discussed.

Keywords: Multiobjective combinatorial optimization, subset selection, ϵ-indicator

1. Introduction

A common heuristic approach to multiobjective optimization is to keep a set of solutions, named *population*, in memory, which is improved iteratively by some operator. In order to maintain a population of fixed cardinality over the run of the heuristic, a *subset selection procedure* chooses a subset of the new population to undergo local changes in the next iteration. Several well-known approaches, such as IBEA [2] and SMS-EMOA [3], use indicator-based measures in the selection process. These indicators measure the quality of the population by assigning it a scalar value. Two of the most common indicators are the hypervolume and ϵ-indicator; we refer to [4, 5, 6] for a more detailed discussion about their properties and usage as subset selection criteria.

Recently, there has been some interest on the design of subset selection procedures that compute an optimal subset and their integration on heuristic approaches. Efficient subset selection approaches are strongly required since they play an important role in the running time of heuristic approaches to multiobjective optimization. Bader [7] describes an $O(n^3)$-time dynamic programming algorithm that finds the optimal subset from a population of n elements with respect to the hypervolume indicator for the biobjective case. Ponte et al. [1] uses the ϵ-indicator within a beam-search algorithm to select a subset from the population that is as *close* as possible to a reference set for the biobjective case. This approach takes $O(mn \log(m+n))$-time, where m and n are the size of the population and the reference set, respectively. However, the correctness of the approach and potential improvements are not discussed in the article.

This note gives a proof of correctness of Ponte's subset selection algorithm, as well some algorithmic improvements on time complexity. Moreover, extensions to larger dimensions are discussed.

2. Notation and Ponte's algorithm

In the following, we use the standard notions of multiobjective optimization [8]. Given two points $p, q \in \mathbb{R}^d_{>0}$, we say that p *dominates* q, or $p \geq q$, if and only if $p^i \geq q^i$, $i = 1, \ldots, d$ with at least one strict

*Corresponding author
Email addresses: dvaz@student.dei.uc.pt (Daniel Vaz), paquete@dei.uc.pt (Luís Paquete), anibal.ponte@estsetubal.ips.pt (Aníbal Ponte)

inequality. If neither $p \geq q$ nor $q \geq p$ holds, then p and q are mutually nondominated.

Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ be two sets of mutually nondominated points in $\mathbb{R}^d_{>0}$, such that no point b in B is dominated by a point a in A. We assume that $a_i^1 < a_j^1$ and $b_i^1 < b_j^1$ implies that $i < j$. The ϵ-indicator is defined as follows:

$$I_\epsilon(A, B) = \max_{b \in B} \min_{a \in A} \epsilon(a, b) \tag{1}$$

where $\epsilon(a, b) = \max\{b^i / a^i\}$, for $i = 1, \ldots, d$. Given a positive integer $k \leq n$, the *ϵ-indicator subset selection problem* consists of finding a set A^* as follows

$$A^* \in \arg\min_{\substack{A' \subseteq A \\ |A'| = k}} I_\epsilon(A', B) \tag{2}$$

Ponte et al. [1] proposes an algorithm for $d = 2$ that solves a sequence of *easy* set covering problems. Therefore, throughout the paper we assume $d = 2$, except when explicitly stated. For a given ϵ value, we formulate the set covering problem as follows. Let $\mathcal{A} = \{1, 2, \ldots, n\}$ be the set of columns and $\mathcal{B} = \{1, 2, \ldots, m\}$ be the set of rows of a zero-one matrix C. For a given ϵ, each entry c_{ij} in matrix C is generated as follows:

$$c_{ij} = \begin{cases} 1 & \text{if } \epsilon(a_j, b_i) \leq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

A cover of C is a set of columns $\mathcal{A}' \subseteq \mathcal{A}$ such that for each row i there exists a column $j \in \mathcal{A}^*$ with $c_{ij} = 1$. The goal is to report whether a k-cover exists, that is, exactly k columns cover all rows of C.

By solving the set covering problem above for different ϵ values, it is possible to solve Problem (2). The overall time complexity derives from three main steps:

- Step 1 (*Preprocessing step*): Sort the mn possible ϵ values;
- Step 2 (*Search step*): Perform binary search on the mn possible ϵ values;
- Step 3 (*Feasibility step*): Find whether a k-cover exists for each ϵ value in Step 2.

The overall time complexity is expressed by $T = T_p + T_s \cdot T_f$ where T_p, T_s and T_f correspond to the time taken by Step 1, 2 and 3, respectively. According to the authors, matrix C of the set covering problem above has the *consecutive ones property* (C1P), that is, the entries with a value of 1 in each row are consecutive [9]. Due to this property, this problem can be solved in $O(m)$ amount of time, after the matrix construction [10]. However, the complexity for Step 3 is $O(mn)$ since it corresponds to the time complexity for the matrix construction. The authors report the following time complexity for each step: $T_p = O(mn \log(m + n))$, $T_s = O(\log(m + n))$, and $T_f = O(mn)$. Therefore, the overall time complexity is $T = O(mn \log(m + n))$.

The authors reduce the number of calls of Step 2 in practice by bounding the number of values of ϵ to test. The lower bound is given by $\epsilon_m = I_\epsilon(A, B)$, which is the minimum ϵ value for which a n-cover exists. The upper bound is $\epsilon_M = \min_{a \in A} I_\epsilon(\{a\}, B)$, which corresponds to the maximum value for which a 1-cover exists. Note that there may not exist a k-cover for $\epsilon = \epsilon_m$ but it always exists for $\epsilon = \epsilon_M$; in the latter case, it is enough to add arbitrary $k - 1$ columns to the 1-cover since they will not increase the value of ϵ. In addition, these bounds do not change the time complexity of Step 2.

3. Correctness

In the following, we show the correctness of the approach of Ponte et al. by proving that matrix C of each set covering problem has C1P.

Proposition 3.1. *There can be no row of matrix C consisting only of zeroes for $\epsilon \geq \epsilon_m$.*

Proof. There always exists a n-cover for $\epsilon \geq I_\epsilon(A, B)$. □

Algorithms	Time complexity
Original algorithm [1]	$O(mn\log(m+n))$
1. Bitonic rows	$O(mn\log(m+n))$
2. Bitonic rows + merge sort	$O(mn\log m)$
3. Bitonic rows + fractional cascading for $m<n$	$O(mn+m^2\log n)$
4. Bitonic rows + fractional cascading for $m>n$	$O(m(n+\log m)\log n)$

Table 1: Summary of the time complexities of the different versions

Proposition 3.2. *Matrix C has C1P for $\epsilon \geq \epsilon_m$.*

Proof. Assume that for an arbitrary row i in matrix C, there exist two non-consecutive ones, i.e. there is a zero in between. Note that this is the only possibility due to Proposition 3.1. Then, there exist points $a_x, a_y, a_z \in A, a_x^1 < a_y^1 < a_z^1$ such that $\epsilon(a_x, b) \leq \epsilon$, $\epsilon(a_y, b) > \epsilon$ and $\epsilon(a_z, b) \leq \epsilon$. As such, since $\epsilon \geq \epsilon(a_x, b) = \max(b^1/a_x^1, b^2/a_x^2)$ then $b^1/a_x^1 \leq \epsilon$. Since $a_x^1 < a_y^1$ if and only if $b^1/a_x^1 > b^1/a_y^1$ then $b^1/a_y^1 \leq \epsilon$. Following a similar reasoning for a_y, a_z, and given that $a_y^2 > a_z^2$ (or a_y would be dominated by a_z), we conclude that $b^2/a_y^2 \leq \epsilon$. Finally, since we have that $\epsilon(a_y, b) = \max(b^1/a_y^1, b^2/a_y^2)$ and both $b^1/a_y^1 \leq \epsilon$ and $b^2/a_y^2 \leq \epsilon$ holds, then $\epsilon(a_y, b) \leq \epsilon$ also holds, which leads to a contradiction. □

4. Algorithmic improvements

4.1. Bitonic rows

Rather than using matrix C, we consider a matrix D with entries $d_{ij} = \epsilon(a_j, b_i)$, $a_j \in A$, $b_i \in B$. Since $\epsilon(a_j, b_i) = \max(b_i^1/a_j^1, b_i^2/a_j^2)$, each row in matrix D is bitonic: as a^1 increases and consequently, a^2 decreases, the value of $\epsilon(a_j, b_i)$ decreases if $b_i^1/a_j^1 > b_i^2/a_j^2$ and increases if $b_i^1/a_j^1 < b_i^2/a_j^2$. Therefore, the bitonic row can be split into two sorted sequences and binary search can be applied to each sequence. As a result, finding the indexes in each of the two sequences where $\epsilon(a_j, b_i)$ is as large as possible, while being at most ϵ can be performed in $O(\log n)$. These are the locations of the first and last entry with the value 1, and we can consider a compressed representation for C, which only stores these two locations for each row. Since we need to find these locations for each of the m lines, we have that $T_f = O(m\log n)$.

4.2. Merge sort

Due to the bitonic nature of the rows in matrix D, it is possible to sort each row in $O(n)$-time by performing a merge of the increasing and decreasing subsequences. This allows to sort the list of possible ϵ values by first sorting each row in $O(mn)$ and then merging all the rows in $O(\log m)$ iterations, building sorted lists of size $2n$ in the first iteration, then $4n$, and so on until the entire list is sorted. Since at each iteration, the size of the merged lists doubles, $O(\log m)$ iterations are needed. Therefore, we have that $T_p = O(mn\log m)$ since the algorithm has to run through the entire list in each iteration. Together with the improvement described in the previous section, we have an overall time complexity of $T = O(mn\log m)$.

4.3. Fractional cascading

A possibility for improving the search step is to use *fractional cascading*, a technique that allows searching in a set of sorted lists without executing binary search in every list. This requires a preprocessing step of $O(mn)$, which consists of creating a new matrix, based on matrix D described in Section 4.1. This transformation is standard and we refer to Chazelle and Guibas [11] for a detailed description (see also its application for computing layers of mutually nondominated points [12]). With this technique, the preprocessing becomes $T_p = O(mn)$. We distinguish two cases: i) $m < n$, which gives a search step with $T_s = O(m + \log n)$; ii) $m > n$, which gives a search step with $T_s = O(n + \log m)$ by using the transpose of the new matrix and performing the search over the rows (columns of the original matrix); since the columns have the same bitonic nature as the rows, the same algorithm above can be applied, but scanning over the

	$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$
$\mathbf{b_1}$	3.0	2.5	5.0
$\mathbf{b_2}$	4.0	8.0	4.0
$\mathbf{b_3}$	5.0	2.5	3.0

Table 2: Matrix D for the proof of Proposition 5.1

columns of the original matrix, when doing the preprocessing step. Since testing each value of ϵ can be done in $O(m \log n)$-time (see Section 4.1), we obtain the overall time complexity $T = O(mn + m^2 \log n)$ for the first case and $T = O(m(n + \log m) \log n)$ for the second case.

4.4. Discussion

Table 1 presents the time complexities of the different versions of the subset selection algorithm. Note that the last three approaches improve the upper bound of the original algorithm. Moreover, the fractional cascading approaches have the best time complexity for each of the two cases. In the context of subset selection procedures within heuristic approaches, n, the size of the population, is kept constant and $m > n$ usually holds [1]. Under this assumption, the fourth improvement is the most favorable in terms of time complexity.

The ϵ-indicator subset selection can be applied to the case where the reference set B is not available and the goal is to find k elements of set A that minimize the ϵ-indicator. This scenario is similar to the one described by Bader [7] with the hypervolume indicator. In our case, it is enough to consider that $B = A$, resulting in an $O(n^2 \log n)$-time approach and improving over the time complexity of the dynamic programming algorithm proposed by Bader. Of particular interest is to know whether a subset selection procedure with $O(n^2)$ time complexity can ever be achieved for any of the two indicators.

5. Extensions

For $d = 2$, matrix C of the underlying set covering problem has C1P. However, this property may not hold for $d > 2$, as we state in Proposition 5.1, which means that the same algorithm cannot be applied to the generic case.

Proposition 5.1. *For $d > 2$, matrix C is not guaranteed to have C1P, even by reordering its rows or columns.*

Proof. Consider, without loss of generality, the case $d = 3$, since for additional dimensions, the vectors may have a fixed value, which does not change the ϵ-indicator and consequently, matrix D. Let $B = \{(3, 5, 2), (4, 4, 8), (5, 3, 2)\}$ and $A = \{(1, 3, 2), (2, 2, 1), (3, 1, 2)\}$. Matrix D is given in Table 2. For $\epsilon = 4$, matrix C is composed of ones, except for the values of the anti-diagonal, which are zero, since the values in matrix D in that positions are larger than ϵ. Therefore, there exist three zeroes, one in each column and one in each row. Therefore, independently of the ordering of the rows or columns, one of the rows will have a zero in the central position, with two non-consecutive ones at the leftmost and rightmost positions. □

Acknowledgments. This work was supported by the bilateral cooperation project "RepSys - Representation systems with quality guarantees for multi-objective optimization problems" founded by the Deutscher Akademischer Austausch Dienst and Conselho de Reitores das Universidades Portuguesas. The authors acknowledge Carlos M. Fonseca, Kathrin Klamroth and Michael Stiglmayr for useful discussions on the main topic of this work.

[1] A. Ponte, L. Paquete, J. Figueira, On beam search for multicriteria combinatorial optimization problems, in: N. Beldiceanu, N. Jussien, E. Pinson (Eds.), Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimization Problems - 9th International Conference, CPAIOR 2012, Nantes, France, May 28 - June 1, 2012, Proceedings, vol. 7298 of *Lecture Notes in Computer Science*, Springer, 307–321, 2012.

[2] E. Zitzler, S. Künzli, Indicator-Based Selection in Multiobjective Search., in: X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. M. Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiño, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings, vol. 3242 of *Lecture Notes in Computer Science*, Springer, 832–842, 2004.
[3] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: Multiobjective selection based on dominated hypervolume, European Journal of Operational Research 181 (3) (2007) 1653–1669.
[4] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. Grunert da Fonseca, Performance Assessment of Multiobjective Optimizers: An Analysis and Review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.
[5] R. Berghammer, T. Friedrich, F. Neumann, Convergence of set-based multi-objective optimization, indicators and deteriorative cycles, Theoretical Computer Science 456 (2012) 2–17.
[6] K. Bringmann, T. Friedrich, Approximating the least hypervolume contributor: NP-hard in general, but fast in practice, Theoretical Computer Science 425 (2012) 104–116.
[7] J. Bader, Hypervolume-Based Search for Multiobjective Optimization: Theory and Methods, Ph.D. thesis, ETH Zurich, Switzerland, 2010.
[8] M. Ehrgott, Multicriteria Optimization, vol. 491 of *Lecture Notes in Economics and Mathematical Systems*, Springer, 2000.
[9] D. Fulkerson, O. Gross, Incidence matrices and interval graphs, Pacific Journal of Mathematics 15 (1965) 835–855.
[10] A. Schöbel, Set covering problems with consecutive ones property, Tech. Rep. 2005-03, Georg-August Universität Göttingen, Institut für Numerische und Angewandte Mathematik, 2005.
[11] B. Chazelle, L. Guibas, Fractional Cascading: I. A Data Structuring Technique, Algorithmica 1 (2) (1986) 133–162.
[12] A. Buchsbaum, M. Goodrich, Three-dimensional layers of maxima, Algorithmica 39 (4) (2004) 275–286.