Output-Sensitive Algorithms for Enumerating the Extreme Nondominated Points of Multiobjective Combinatorial Optimization Problems

Fritz Bökler* and Petra Mutzel

Department of Computer Science, TU Dortmund, Germany
{fritz.boekler,petra.mutzel}@tu-dortmund.de

Abstract. This paper studies output-sensitive algorithms for enumeration problems in multiobjective combinatorial optimization (MOCO). We develop two methods for enumerating the extreme points of the Pareto-frontier of MOCO problems. The first method is based on a dual variant of Benson's algorithm, which has been originally proposed for multiobjective linear optimization problems. We prove that the algorithm runs in output polynomial time for every fixed number of objectives if the weighted-sum scalarization can be solved in polynomial time. Hence, we propose the first algorithm which solves this general problem in output polynomial time. We also propose a new lexicographic version of the dual Benson algorithm that runs in incremental polynomial time in the case that the lexicographic optimization variant can be solved in polynomial time. As a consequence, the extreme points of the Pareto-frontier of the multiobjective spanning tree problem as well as the multiobjective global min-cut problem can be computed in polynomial time for a fixed number of objectives. Our computational experiments show the practicability of our improved algorithm: We present the first computational study for computing the extreme points of the multiobjective version of the assignment problem with five and more objectives. We also empirically investigate the running time behavior of our new lexicographic version compared to the original algorithm.

1 Introduction

In practical optimization, we often deal with problems having more than one objective. Unlike in single objective optimization, there usually does not exist a single optimal value and we usually have many solutions which are incomparable. If we agree that we will always prefer a solution over all solutions that are worse in all objectives then we can use this relation as a partial order on the solutions, called the *Pareto-dominance*. More precisely, we say a vector $a \in \mathbb{R}^d$ *dominates* a vector $b \in \mathbb{R}^d$ or $a \preceq b$ if $a \leq b$ (componentwise) and $a \neq b$. Analogously, for

* The author has been supported by the Bundesministerium für Wirtschaft und Energie (BMWi) within the research project "Bewertung und Planung von Stromnetzen" (promotional reference 03ET7505) and by DFG GRK 1855 (DOTS).

N. Bansal and I. Finocchi (Eds.): ESA 2015, LNCS 9294, pp. 288–299, 2015.
DOI: 10.1007/978-3-662-48350-3_25

two solutions $x, y \in \mathcal{X}$ and an objective function $c : \mathcal{X} \to \mathbb{R}^d$ we say $x \preceq y$ if $c(x) \preceq c(y)$. The set of all minimal solutions with respect to this partial order is called the set of *efficient* or *Pareto-optimal* solutions; the corresponding image under the objective function is called the set of *nondominated points* or *Pareto-frontier*.

In this spirit, we can define *multiobjective combinatorial optimization (MOCO) problems* consisting of a base set $\mathcal{A}$, a set of feasible solutions $\mathcal{S} \subseteq 2^{\mathcal{A}}$ and an objective function $c : \mathcal{A} \to \mathbb{Q}^d$. The cost of a solution $x \in \mathcal{S}$ thus is $c(x) := \sum_{a \in x} c(a)$. The goal is to find for all points y of the Pareto-frontier one solution x such that $c(x) = y$. This notion is motivated by the multicriteria decision making community: Two solutions which are mapped to the same image by the objective function are supposed to be essentially the same object.

One serious computational issue of many MOCO problems is that the size of the Pareto-frontier can be large in the worst case, i.e., exponential in the input size. Then again, the output being exponential is also true for many other enumeration problems and has been addressed by the theory of output-sensitive complexity. Johnson, Papadimitriou and Yannakakis [13] provide a summary of these complexity notions: An algorithm runs in *output polynomial time* (originally referred to as *polynomial total time*) if its running time can be bounded by a polynomial in the input and the output size. Besides the total running time, a usually more interesting property is the delay of an enumeration algorithm. Let N be the number of elements to output. We say the 0-th delay is the running time prior to the first output of a solution, the k-th delay is the running time between the output of the k-th and $(k+1)$-th solution, and the N-th delay is the running time after the last output until the termination of the algorithm. An enumeration algorithm runs in *polynomial (time) delay* if all delays are bounded by a polynomial in the input size. To relax this notion a bit, an algorithm runs in *incremental polynomial time*, if the k-th delay is bounded by a polynomial in the input and k. One purpose of this paper is to encourage a line of research which applies these complexity notions to multiobjective optimization problems and—surprisingly enough—there is only one other paper which considers this [14].

Moreover, in a recent work by Röglin and Brunsch [5], it is shown that in a smoothed analysis setting, the expected size of the Pareto-frontier of a MOCO problem of n variables and d objectives is $\mathcal{O}(n^{2d}\Phi^d)$ for every perturbation parameter $\Phi \geq 1$. This result gives reason to believe that pursuing the goal to find the entire Pareto-frontier might still be practical as long as the number of objectives is not too large.

A very promising approach to solve MOCO problems is the *two-phase method* (cf. e.g., [7]). In this method, the Pareto-frontier is partitioned into two sets: *extreme* nondominated and *nonextreme* nondominated points. (Exact definitions of these sets will be given later.) In a first phase, we compute the extreme nondominated points. In the second phase we can exploit this knowledge to find the remaining nondominated points. The two-phase method is motivated by the observation that it is often **NP**-hard even to decide if there exists a solution which dominates a given point $y \in \mathbb{Q}^d$. But at least in the biobjective

case, if the lexicographic variant can be solved in polynomial time, then we can enumerate the set of extreme nondominated points in incremental polynomial time by using standard methods like the dichotomic approach [3]. Although the problem of finding the extreme nondominated points of a MOCO problem seems to be solved in the biobjective case, it is an open problem to enumerate these points and corresponding solutions for problems with an arbitrary number of objectives in reasonable time. Ehrgott and Gandibleux address this problem in their survey paper from 2005 [7] as a "first step to an application of the two phases method in three or more criteria MOCO". The main concern of this paper will be how to compute the set of extreme nondominated points in output polynomial running time in theory and practice.

For $d = 2$, finding the extreme points of the Pareto-frontier is equivalent to the combinatorial parametric optimization problem. See, e.g., [1] for a parametric version of the minimum spanning tree problem. Thus, the problem we consider here is a natural generalization of this very well known class of problems.

Previous Work. There exist only a few other methods to enumerate the extreme nondominated points for MOCO problems with more than two objectives. Przybylski, Gandibleux and Ehrgott [17] propose a method to find these points for general multiobjective integer linear programming problems for which the ideal point exists. The algorithm is theoretically capable of computing the extreme nondominated points for an arbitrary number of objectives and the authors provide deep improvements in the case of three objectives. They also conducted a set of experiments on three-objective assignment and knapsack problems showing a very decent running time. Özpeynirci and Köksalan [15] suggest a method to compute the set of extreme nondominated points based on the dichotomic approach which is usually utilized in the biobjective case. The authors also present experiments on several MOCO problems with three and four objectives. Both of the above works do not provide running time guarantees for their algorithms.

In [14], the authors propose an algorithm that enumerates all efficient spanning trees which are a solution to a weighted-sum of the objectives with polynomial delay. The algorithm bases on the reverse search by Avis and Fukuda [4] to output the efficient vertices of the spanning tree polytope. A note is here in order, because the algorithm follows a different model where all efficient solutions are sought. It is possible that there exist many spanning trees which are mapped to the same point in the objective space. Consider a complete graph with n vertices where all edges are mapped to the $\mathbf{1}$-vector in $\mathbb{R}^2$ by the objective function. Then each of the n^{n-2} spanning trees is efficient, but the Pareto-frontier consists of only one point.

Another branch of research which tackles the problem of an exponential sized Pareto-frontier of MOCO problems was raised by a paper by Papadimitriou and Yannakakis [16]. In this work and in many papers that followed, an ε-Pareto set, i.e., a subset S of the solution set such that for each point y of the Pareto-frontier there is one solution $x \in S$ such that $c(x) \leq (1 + \varepsilon)y$, is computed. In [16], it is also proven that for each $\varepsilon > 0$ and fixed number of objectives there exist such an ε-Pareto set of size polynomial in the input size and $\frac{1}{\varepsilon}$.

Our Contributions. A well known method to find extreme nondominated points (or more general supported points, see below) in the first phase of the two-phase method for a MOCO problem $(\mathcal{A}, \mathcal{S}, c)$ is to optimize the weighted-sum scalarization: $(P_1(\ell)) : \min\{\ell^T c(x) \mid x \in \mathcal{S}\}$ for some $\ell \in \mathbb{Q}^d, \ell > 0$. On one hand, this problem is as easy as the single objective version of the problem, as long as the encoding lengths of the components of ℓ are not too large. (Considering that ℓ is not part of the original input.) On the other hand, we do not easily know how these ℓ need to be chosen to find all obtainable points of the Pareto-frontier.

The set of solutions which can be obtained by this weighted-sum method are called *supported (efficient) solutions*. A point of the Pareto-frontier which corresponds to a supported solution is called a *supported (nondominated) point*. A geometric view on the supported part of the Pareto-frontier can be acquired by looking at the convex hull of $\mathcal{Y} := c(\mathcal{S})$. A point y of the Pareto-frontier is thus supported iff y is a point on the boundary of $\operatorname{conv}\mathcal{Y}$. Moreover, we call a supported point y an *extreme (nondominated) point*, if y is an extreme point of $\operatorname{conv}\mathcal{Y}$. The set of extreme nondominated points will be denoted as $\mathcal{Y}_X$. Our concern in this paper will thus be to enumerate the set $\mathcal{Y}_X$.

In general, we can compute the extreme nondominated points of $\operatorname{conv}\mathcal{Y}$ by enumerating the extreme nondominated points of the *multiobjective linear program*

$$(\text{MOLP}) : \min\{c(x) \mid x \in \operatorname{conv}(\chi(\mathcal{S}))\}, \tag{1}$$

where $\chi(\mathcal{S})$ denotes the characteristic vectors of the solutions of our MOCO problem for a fixed ordering of the variables. In Sec. 3, we will conduct a running time analysis of a recently proposed algorithm for MOLP. We will prove that it can efficiently find the extreme points of the Pareto-frontier of an MOLP if the ideal point exists. Luckily, in the case in which we derive an MOLP from a MOCO problem, the ideal point exists iff the problem has a solution. But indeed, it might be hard to construct the MOLP, the number of facets of the feasible set might be large or it might have a large encoding length. We will also show that it suffices to have a polynomial time algorithm for problem $(P_1(\ell))$ without constructing the MOLP explicitly.

Theorem 1. *For every MOCO problem P with a fixed number of objectives, the set of extreme nondominated points of P can be enumerated in output polynomial time if we can solve the weighted-sum scalarization of P in polynomial time.*

Subsequently in Sec. 4, we will also suggest an improvement of this algorithm to get a better running time at the expense of needing to solve the lexicographic version of the MOCO problem $(\text{lex-}P_1(\ell)) : \operatorname{lexmin}\{(\ell^T c(x), c_1(x), \ldots, c_d(x)) \mid x \in \mathcal{S}\}$:

Theorem 2. *For every MOCO problem P with a fixed number of objectives, the set of extreme nondominated points of P can be enumerated in incremental polynomial time if we can solve the lexicographic version of P in polynomial time.*

For many classical problems in combinatorial optimization this is not a restriction. For example in the case of shortest path, minimum spanning tree or the assignment problems the lexicographic variant can be solved in polynomial time. In general, if we have a compact LP formulation for a weighted-sum scalarization, then we can solve the lexicographic variant in polynomial time. This is a direct consequence of Lemma 6 in Sec. 4.

If we apply Theorem 1 to the multiobjective spanning tree problem, we immediately obtain an algorithm with a polynomial upper bound on the running time with respect only to the input size to find the extreme nondominated points for each fixed number of objectives. This is well known in the biobjective (or parametric) case, but it is a new result for the general case. It bases on the well known fact that for this problem there exists at most $\mathcal{O}(m^{2(d-1)})$ extreme nondominated points [10]. We obtain a similar result for the multiobjective global min-cut problem, as it has been shown in [2] that the parametric complexity is polynomial in the input size for any fixed number of objectives. To show that these results are not mere theoretical, we implemented the algorithm and a lexicographic oracle for the multiobjective version of the assignment (or minimum weight bipartite matching) problem. The results of this study and a comparison to the method from [15] are presented in Sec. 5. They show that our algorithm is capable of finding the extreme nondominated points of moderately sized instances with up to six objectives. To the best of our knowledge this is the first computational study for five and six objectives.

Moreover, we compare the lexicographic variant of the algorithm to the original algorithm. Since in theory we are able to improve the running time bound, we investigate if this is also true in practice.

2 Theoretical Preliminaries

By $[n]$ we denote the set $\{1, \dots, n\}$ and $||x||_1$ will be the 1-*norm* of $x \in \mathbb{R}^d$, i.e., $||x||_1 := \sum_{i=1}^{d} |x_i|$. The nondominated subset of a set of points $M \subseteq \mathbb{R}^d$ is $\min M := \{x \in M \mid x \text{ is nondominated in } M\}$. The *Minkowski sum (product)* of two sets $A, B \subseteq \mathbb{R}^d$ is the set $A + B := \{a + b \mid a \in A, b \in B\}$ ($A \cdot B := \{a \cdot b \mid a \in A, b \in B\}$).

The *multiobjective linear programming problem* (MOLP) is the problem to find for matrices $A \in \mathbb{Q}^{m \times n}, C \in \mathbb{Q}^{d \times n}$ and a vector $b \in \mathbb{Q}^m$ an extreme point representation of the Pareto-frontier $\min\{Cx \mid Ax \geq b\}$. To work on MOLP, it is convenient to define the *feasible set* $P := \{x \in \mathbb{R}^n \mid Ax \geq b\}$ and the *upper image* $\mathcal{P} := C \cdot P + \mathbb{R}^d_{\geq}$, where $\mathbb{R}^d_{\geq} := \{x \in \mathbb{R}^d \mid x \geq 0\}$. It is well known that the extreme points of $\mathcal{P}$ are exactly the extreme points of the Pareto-frontier of the corresponding MOLP (cf. [8]). We will write $\operatorname{vert} M$ to denote the set of extreme points of a polyhedral set M and thus, our concern will be to enumerate $\operatorname{vert} \mathcal{P}$, i.e., the extreme points of $\mathcal{P}$.

We define the *normalized weighting vectors* to be the set $W_d^0 := \{\ell \in \mathbb{R}^d_{\geq} \mid ||\ell||_1 = 1\}$. The *weighted-sum linear program w.r.t.* $\ell \in W_d^0$ of a given MOLP is the parametric linear program $P_1(\ell) : \min\{\ell^T Cx \mid Ax \geq b\}$. The *ideal point*

of an MOLP is then defined as being the point $y^I := (\min P_1(e_i))_{i\in[d]}$, where e_i denotes the i-th unit vector in $\mathbb{R}^d$. Note that an ideal point does not exist for every MOLP.

3 Dual Variant of Benson's Algorithm

The dual variant of Benson's algorithm solves MOLP in the sense that it computes the extreme points of a polyhedron which is a geometric dual polyhedron to $\mathcal{P}$. The algorithm in its original version requires the existence of an ideal point, which we will assume in the following and is always the case in MOCO problems. It was only recently proposed in [8] and got its background theory from [12]. In the next section, we will consider the algorithm as it was given in [8, 11]. We will first give some background theory about the geometric dual polyhedron the algorithm computes and subsequently, describe the algorithm in two levels of detail. The section concludes with its running time analysis in the sense of output-sensitive complexity.

The Geometric Dual Polyhedron. In [12], Heyde and Löhne define a dual polyhedron, or *lower image*, $\mathcal{D}$ to the upper image $\mathcal{P}$ of an MOLP which we will define now.

Since our weight vectors ℓ of the weighted-sum problem are always normalized, i.e., $||\ell||_1 = 1$, it suffices to consider $\ell_1, \dots, \ell_{d-1}$ and calculate ℓ_d when needed. For ease of notation we define for $v \in \mathbb{R}^d : \lambda(v) := (v_1, \dots, v_{d-1}, 1 - \sum_{i=1}^{d-1} v_i)$. Then, we consider the dual problem of the weighted-sum LP, which is $(D_1(\ell)) : \max\{b^T u \mid u \in \mathbb{R}^m_{\geq}, A^T u = C^T \ell\}$. The dual polyhedron $\mathcal{D}$ now consists for all possible vectors $\ell \in W_d^0$ and solutions u to $D_1(\ell)$ of the vectors $(\ell_1, \dots, \ell_{d-1}, b^T u)$. Thus, $\mathcal{D} := \{(\ell_1, \dots, \ell_{d-1}, b^T u) \in \mathbb{R}^d \mid \ell \in W_d^0, u \in \mathbb{R}^m_{\geq}, A^T u = C^T \ell\}$. Following LP duality theory, for each point y on the upper boundary of this polyhedron y_d is also the optimal value of $P_1(\lambda(y))$. To take the notion of the upper boundary to a more formal level, we define the $\mathcal{K}_d$-maximal subset of a set $M \subseteq \mathbb{R}^d$, where $\mathcal{K}_d := \{(0, \dots, 0, y) \in \mathbb{R}^d \mid y \geq 0\}$: A point $y \in M$ is said to be $\mathcal{K}_d$-*maximal* in M if $(y + \mathcal{K}_d) \cap M = \{y\}$. The subset of $\mathcal{K}_d$-maximal points of M is written as $\max_{\mathcal{K}_d} M$.

In [12], it is proven, that the dual polyhedron can be characterized by $\mathcal{D} = \{x \in \mathbb{R}^d \mid \forall y \in \operatorname{vert} \mathcal{P} : \psi(y)^T x \geq -y_d, \lambda(x) \geq 0\}$, where $\psi(y) := (v_1 - v_d, \dots, v_{d-1} - v_d, -1)$. In other words, apart from the inequalities $\lambda(x) \geq 0$, we can describe the polyhedron as an intersection of halfspaces $\{x \in \mathbb{R}^d \mid \psi(y)^T x \geq -y_d\}$ for each extreme point y of the upper image $\mathcal{P}$. Further, Heyde and Löhne also prove that each of these inequalities defines a facet. Thus, we can solve the original MOLP by enumerating the facets of $\mathcal{D}$. While the dual algorithm originally enumerates the extreme points of $\mathcal{D}$, we change the exposition accordingly.

Algorithm Description. We will follow [8] in describing the geometric dual algorithm and use ideas of [11]. Proofs of correctness and finiteness can be found in both places. A formal description of the entire algorithm can be found in

Algorithm 1. Dual Variant of Benson's Outer Approximation Algorithm

Require: Matrices A, C, and vector $b : P \neq \emptyset$ and $\exists y \in \mathbb{R}^d : y + C \cdot P \subseteq \mathbb{R}^d_{\geq}$
Ensure: List R of pairs (x, y) for all $y \in \operatorname{vert} \mathcal{P}$ and some $x \in P$ such that $Cx = y$
1: Find solution x of $P_1(e_1)$ and set $y \leftarrow Cx$
2: $L \leftarrow \{x \in \mathbb{R}^d \mid \lambda(x) \geq 0, \psi(y)^T x \geq -y_d\}$ ▷ Initial polyhedron
3: $M \leftarrow$ Extreme points of L ▷ Perform a vertex enumeration
4: **while** $M \neq \emptyset$ **do**
5: pick one $v \in M$, $M \leftarrow M \backslash \{v\}$
6: $x \leftarrow$ optimal solution to $P_1(\lambda(v))$ and $y \leftarrow Cx$ ▷ Shoot ray straight down
7: **if** $\lambda(v)^T y < v_d$ **then** ▷ Not an extreme point of $\mathcal{D}$
8: $L \leftarrow L \cap \{x \in \mathbb{R}^d \mid \psi(y)^T x \geq -y_d\}$ ▷ Add new inequality
9: $M \leftarrow$Extreme points of L ▷ Perform a vertex enumeration
10: $R \leftarrow L \cup \{(x, y)\}$ ▷ Add new candidate extreme point of $\mathcal{P}$
11: Remove redundant entries from R

Alg. 1. From a high-level perspective, the algorithm works as follows: First, the algorithm constructs a polyhedron containing $\mathcal{D}$ (lines 1 to 3). Then, in each iteration, it picks one new extreme point v of the current intermediate polyhedron and shoots a ray into the polyhedron $\mathcal{D}$ (lines 5 and 6). We can shoot a ray in the direction of $-\mathcal{K}_d$ by finding an optimal solution x of $P_1(\lambda(v))$ with value vector $y = Cx$. Either, we discover that v is an extreme point of $\mathcal{D}$, if $\lambda(v)^T y = v_d$ and we proceed to the next iteration. Or, v is not an extreme point, which we see if $\lambda(v)^T y < v_d$ or in other words, the weighted optimal value is smaller than the value represented by v. Then, the algorithm computes a face defining inequality which separates v from $\mathcal{D}$. Because of geometric duality, we can use the new inequality $\psi(y)^T x \geq -y_d$. In lines 8 and 9, the algorithm intersects the current polyhedron with the halfspace corresponding to this inequality. Additionally, it saves y as a candidate for an extreme nondominated point in line 10. This repeats until all extreme points have been confirmed to be part of $\mathcal{D}$. In the end, we still have to remove redundant pairs from the set of candidate extreme nondominated points (see line 11).

Running Time Analysis. The key insight to the running time of the algorithm is that the vertex enumeration steps are performed in the ordinarily much smaller domain of the polyhedron $\mathcal{D}$, which is of dimension d. Additionally, the number of inequalities we enumerate in the process of the algorithm is at most the number of $\mathcal{K}$-maximal faces of $\mathcal{D}$ which—by the geometric duality theorem of [12]—is exactly the number of faces of $\mathcal{P}$. The number of faces of $\mathcal{P}$ can be bounded by reducing the polyhedron to a polytope and using the asymptotic upper bound theorem [18]. Let thus v_e be the number of extreme points of $\mathcal{P}$.

Lemma 3. *Let d be fixed, the number of faces of $\mathcal{P}$ is at most $\mathcal{O}({v_e}^{\lfloor \frac{d}{2} \rfloor})$.*

This shows that the number of faces of $\mathcal{D}$ and thus the number of inequalities the algorithm computes does not exceed $\mathcal{O}({v_e}^{\lfloor \frac{d}{2} \rfloor})$. To compute the extreme points of the intermediate polyhedra, we can use the asymptotic optimal algorithm for fixed d by Chazelle [6].

Regarding encoding length, there is a subtle problem we need to address. The running times of the known polynomial-time algorithms to compute a solution to the weighted-sum LPs, e.g., the ellipsoid method, depend on the largest number in the input. One potential problem is that the weighted-sum LP not only consists of numbers of the input MOLP, but also of the weight vector ℓ which is recomputed in the process of the algorithm. We can prove that these lengths are not too large by using the fact that the weights are computed from the intermediate extreme points we get. These points are solutions to linear systems of the face defining inequalities we find, which in turn are computed from the extreme points of the upper image $\mathcal{P}$. These extreme points of $\mathcal{P}$ are again linear images of extreme points in the decision space and it is well known that the encoding length of these extreme points can be bounded by $\mathcal{O}(\text{poly}(n, L))$, where L is the encoding length of the largest number in A and thus independent of ℓ.

Lemma 4. *The encoding length of an intermediate extreme point is bounded by* $\mathcal{O}(\text{poly}(d, n, L))$.

This concludes the running time analysis and we can give the running time in the following theorem. We will be able to significantly improve on the d^2 in the exponent in the next section.

Theorem 5. *Let v_e be the number of vertices of $\mathcal{P}$ and d be fixed. Then, Algorithm 1 has a running time bounded by* $\mathcal{O}({v_e}^{\lfloor \frac{d}{2} \rfloor}(\text{poly}(n, m, L) + {v_e}^{\frac{d^2}{4}} \log v_e) + \text{poly}(n, v_e, L))$.

To arrive at Theorem 1, we can solve problem (1). Problem $(P_1(\ell))$ is then equivalent to solving a single linear objective over the feasible set. Instead of constructing the LP explicitly, we can solve the weighted-sum scalarization of the combinatorial optimization problem. Since the encoding length of the weights we use can be bounded by a polynomial in the original input, an algorithm running in weakly polynomial time also suffices to prove the claim.

4 Lexicographic Dual Benson

One serious drawback of the algorithm is that we might enumerate many redundant supporting hyperplanes. This is especially a problem since the number of vertex enumeration steps depends on the number of supporting inequalities we find. Also the number of redundant extreme points we compute depends heavily on this quantity. Hence, it is very much desirable to enumerate facet supporting hyperplanes only. This is the motivation to propose a lexicographic variant of the dual Benson algorithm which we call the lexicographic dual Benson algorithm and will be the subject of this section.

As already stated, in [12], Heyde and Löhne prove that every inequality $\psi(y)^T x \geq -y_d$ is facet defining iff y is an extreme point of $\mathcal{P}$. So it suffices to find only extreme points of $\mathcal{P}$ in lines 1 and 6. We observe that this can be accomplished by computing an optimal solution x of $P_1(\lambda(v))$ with lexicographic minimal $y := Cx$: The set of optimal solutions M of $P_1(\lambda(v))$ is mapped by the

objective function to a face F of $\mathcal{P}$ such that $C \cdot M = F$. If we search for a lexicographic minimal Cx for $x \in M$, we will find a lexicographic minimal point of F. Since F is a polyhedron and assuming the ideal point of the MOLP exists, we will always arrive at an extreme point of F and thus of $\mathcal{P}$.

We define lex-$P_1(\ell)$: lexmin$\{(\ell^T Cx, c_1 x, \ldots, c_d x) \mid Ax \geq b\}$, where $c_1, \ldots, c_d$ are the rows of C and change lines 1 and 6 accordingly. Moreover, the points we add in line 10 are already vertices of $\mathcal{P}$ and we can skip line 11 and do not need to remove redundant entries anymore. We still have to prove that we can solve this lexicographic LP in polynomial time, but we omit the proof because of space restrictions.

Lemma 6. *A solution to a lexicographic LP can be computed in polynomial time.*

Both of these observations are a game changer in this running time analysis and we arrive at the following theorem.

Theorem 7. *Let v_e be the number of vertices of $\mathcal{P}$ and d be fixed. The lexicographic dual Benson algorithm has a running time of $\mathcal{O}(v_e^{\lfloor \frac{d}{2} \rfloor}(\text{poly}(n, m, L) + v_e \log v_e))$.*

This is a significant improvement over Theorem 5, since we eliminate the term having d^2 in the exponent. Moreover, we are now able to bound the delay of the algorithm. In the original algorithm this was not possible since it could take a large number of iterations until the $(d+1)$-th extreme point is found.

Theorem 8. *Let d be fixed. For the lexicographic dual Benson algorithm the k-th delay is in $\mathcal{O}(k^{\lfloor \frac{d}{2} \rfloor} \text{poly}(n, m, L))$.*

Analogously to Theorem 1, we want to use this algorithm to find extreme nondominated points of MOCO problems. Instead of constructing the corresponding lexicographic LP, which is equivalent to solving a lexicographic objective function over the set of feasible points of problem (1), we can solve the lexicographic version of the MOCO problem. Hence, exchanging the lexicographic LP oracle by an algorithm which computes a lexicographic optimal solution of the MOCO problem suffices to prove Theorem 2.

5 Computational Study

We investigated the practical aspects of the lexicographic dual Benson algorithm. To achieve this, we conducted a comparison with the method from [15], which is to the best of our knowledge the only practical method to compute the extreme points of MOCO problems with four and more objectives. Both implementations were tested on instances of the multiobjective assignment (or minimum weight bipartite matching) (MO-A) problem. The MO-A problem is often used as a benchmark problem in the biobjective case, but is also used in the computational studies of the approaches existing for three and four objectives [15, 17].

Table 1. Computational results on multiobjective assignment instances with d objectives and n resources. Instances with an * were taken from [15].

		Lexicographic Dual Benson				**OK10 implementation**				
		Running Time [s]		$\lvert\mathcal{Y}_X\rvert$		Running Time [s]		$\lvert\mathcal{Y}_X\rvert$		no. solved
d	n	Median	MAD	Median	MAD	Median	MAD	Median	MAD	#
3	10*	0.01	0.002	31.5	4.448	0.03	0.010	31.5	4.448	20
	20*	**0.11**	0.014	150.5	17.050	6.31	1.491	150.5	17.050	20
	30*	**0.70**	0.120	368.5	51.150	160.24	57.858	368.5	51.150	20
	40	**2.57**	0.195	709.0	51.150	1660.63	542.646	709.0	51.150	20
	80	67.22	4.442	2819.0	185.325	—	—	—	—	0 (0/20)
	150	1350.28	87.874	9626.0	374.357	—	—	—	—	0 (20/0)
4	10*	**0.06**	0.019	102.5	26.687	3.29	2.447	102.5	26.687	20
	15	0.33	0.079	453.5	97.852	(253.35)	(105.681)	(347.0)	(14.826)	7 (13/0)
	30	12.97	1.824	3646.0	444.039	—	—	—	—	0 (20/0)
	70	1978.68	269.907	48667.0	5777.692	—	—	—	—	0 (20/0)
5	8	0.22	0.131	125.5	34.100	(29.91)	(36.378)	(124.0)	(31.876)	18 (2/0)
	14	33.30	16.416	1228.0	313.570	—	—	—	—	0 (20/0)
	20	1055.82	252.966	5052.5	699.787	—	—	—	—	0 (20/0)
6	6	0.16	0.094	75.0	20.015	(39.01)	(50.977)	(73.0)	(17.791)	18 (2/0)
	8	1.73	1.096	228.0	54.856	(159.25)	(45.658)	(164.5)	(20.015)	2 (18/0)
	12	213.95	159.990	1798.0	549.303	—	—	—	—	0 (20/0)

Though, the algorithm in [15], to which we will refer to as OK10 algorithm, is not easily implemented efficiently. Instead of using points with large encoding length on the axes, we use projective points at infinity to reduce the numerical inconsistencies which occurred in the experimental setting in [15]. Nevertheless, the implementation still misses some extreme nondominated points, but does find more than the original implementation. To compute optimal assignment solutions in the OK10 algorithm, we use an implementation of the Hungarian method.

The experiments were performed on an Intel Core i7-3770, 3.4 GHz and 16 GB of memory running Ubuntu Linux 12.04. The algorithms were implemented in C++ using double precision arithmetic and compiled using LLVM 3.3. To compute the extreme points of the intermediate polyhedra, we implemented a version of the Double Description method with full adjacency information in the case of $d \in \{3, 4\}$ and for $d > 4$, we used the CDD library [9]. To find lexicographic minimal assignments for the lexicographic dual Benson algorithm, we implemented a lexicographic version of the Hungarian method.

The computational study for computing the set of all extreme nondominated points of MO-A instances in [15] uses a series of 20 randomly generated instances. The sizes of the instances vary from 10 up to 30 resources and the integer objective function coefficients were uniformly drawn in $[1, 20]$. For our experiments, we have taken the instances from [15] for $d \in \{3, 4\}$ and additionally generated similar instances with more resources and objectives.[1] We enforced two kinds of limits on the computational experiments. That is, a memory limit of 16 GB and a computation time limit of one hour should not be exceeded.

[1] All instances are available at `https://ls11-www.cs.tu-dortmund.de/staff/boekler/moco-instances`

Table 2. Comparison of the dual Benson implementation with and without lexicographic oracle on multiobjective assignment instances with $d = 5$

	Running Times $\mathcal{T}_{\text{lex}}/\mathcal{T}$						**Points found lex / no lex**	
	Hungarian algorithm		VE		Total			
n	Median	MAD	Median	MAD	Median	MAD	Mean	σ
20	1.114	0.0301	1.000	0.0091	1.002	0.0086	1.000	0.0003
22	1.118	0.0221	1.000	0.0046	1.000	0.0045	1.000	0.0002
24	1.130	0.0110	0.999	0.0055	1.000	0.0055	1.000	<0.0001
26	1.126	0.0170	0.999	0.0044	0.999	0.0045	1.000	<0.0001

In Table 1 we can see some selected results including the median and median absolute deviation (MAD) of the running times and numbers of extreme points found. We observe that the lexicographic dual Benson implementation is able to solve all instances in the given limits. The implementation of the OK10 algorithm is only able to solve very small instances with three and four objectives. In all other cases there are instance classes where the implementation can not solve all of the instances, prohibiting a statistical analysis. Nevertheless, we give the median and MAD in parentheses if the OK10 implementation was not able to solve all instances.

In the last column of Table 1, we can see the number of instances the OK10 implementation was able to solve. In parentheses we see the number of instances that could not be solved due to the memory and time limit, respectively. We see that in most cases memory was the limiting factor. This is not surprising, since for every new point d new states are introduced and many survive the pruning steps. In the cases where the OK10 implementation is able to solve all instances, we see that the lexicographic dual Benson implementation is up to a factor of 640 times faster.

In the second set of experiments, we compare the practical performance of the dual Benson algorithm when using our theoretical improvements from Sec. 4 to the original variant. The same instances as in the previous experiments were used; we present the experiments with five objectives. Table 2 displays the medians and MADs of the quotients of the lexicographic variant over the non-lexicographic variant. The table shows these statistics for the cumulated running time of the Hungarian algorithm, the vertex enumeration (VE) and the total time. In addition, the last column of Table 2 displays the mean and standard deviation of the quotient of the number of points found by both algorithms. Median and MAD are in all cases 1 and 0, respectively.

We observe that the running times are very similar. The quotients of the total running time medians are very close to 1. On one hand, the vertex enumeration is only slightly faster when using a lexicographic oracle. On the other hand, the cumulated time of the lexicographic oracle is always slower than the time of the original Hungarian method. Of course, the vertex enumeration dominates the total running time, but we also observe that it does not happen too often that redundant inequalities are found.

We can also observe that the medians of the total running time quotients seem to shrink when increasing the number of ressources. In order to observe if this trend continues, we need to test much larger instances which is not possible

with the current implementation, because of running times of already more than 12 hours on instances with 26 ressources.

References

1. Agarwal, P.K., Eppstein, D., Guibas, L.J., Henzinger, M.R.: Parametric and kinetic minimum spanning trees. In: IEEE FoCS, pp. 596–605 (1998)
2. Aissi, H., Mahjoub, A.R., McCormick, S.T., Queyranne, M.: A strongly polynomial time algorithm for multicriteria global minimum cuts. In: Lee, J., Vygen, J. (eds.) IPCO 2014. LNCS, vol. 8494, pp. 25–36. Springer, Heidelberg (2014)
3. Aneja, Y.P., Nair, K.P.K.: Bicriteria transportation problem. Management Science 25(1), 73–78 (1979)
4. Avis, D., Fukuda, K.: A pivoting algorithm for convex hulls and vertex enumeration of arrangements of polyhedra. Discrete and Computational Geometry 8(1), 295–313 (1992)
5. Brunsch, T., Röglin, H.: Improved smoothed analysis of multiobjective optimization. In: ACM SToC, pp. 407–426 (2012)
6. Chazelle, B.: An optimal convex hull algorithm in any fixed dimension. Discrete Computational Geometry 10, 377–409 (1993)
7. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spektrum 22, 425–460 (2000)
8. Ehrgott, M., Löhne, A., Shao, L.: A dual variant of Benson's "outer approximation algorithm" for multiple objective linear programming. Journal of Global Optimization 52, 757–778 (2012)
9. Fukuda, K., Prodon, A.: Double description method revisited. In: Deza, M., Manoussakis, I., Euler, R. (eds.) CCS 1995. LNCS, vol. 1120, pp. 91–111. Springer, Heidelberg (1996)
10. Ganley, J.L., Golin, M.J., Salowe, J.S.: The multi-weighted spanning tree problem. In: Li, M., Du, D.-Z. (eds.) COCOON 1995. LNCS, vol. 959, pp. 141–150. Springer, Heidelberg (1995)
11. Hamel, A.H., Löhne, A., Rudloff, B.: Benson type algorithms for linear vector optimization and applications. arXiv:1302.2415 [math.OC] (July 2013)
12. Heyde, F., Löhne, A.: Geometric duality in multiple objective linear programming. SIAM Journal of Optimization 19(2), 836–845 (2008)
13. Johnson, D.S., Yannakakis, M., Papadimitriou, C.H.: On generating all maximal independent sets. Information Processing Letters 27, 119–123 (1988)
14. Okamoto, Y., Uno, T.: A polynomial-time-delay and polynomial-space algorithm for enumeration problems in multi-criteria optimization. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 609–620. Springer, Heidelberg (2007)
15. Özpeynirci, Ö., Köksalan, M.: An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. Management Science 56(12), 2302–2315 (2010)
16. Papadimitriou, C.H., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: IEEE FoCS, pp. 86–92 (2000)
17. Przybylski, A., Gandibleux, X., Ehrgott, M.: 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 (2010)
18. Seidel, R.: The upper bound theorem for polytopes: an easy proof of its asymptotic version. Computational Geometry 5, 115–116 (1995)