An outer approximation algorithm for multiobjective mixed-integer linear programming

Fritz Böcker∗1, Sophie N. Parragh†2, Markus Sinnl‡2, and Fabien Tricoire§3

1Institute of Computer Science, Osnabrück University, Osnabrück, Germany
2Institute of Production and Logistics Management, Johannes Kepler University Linz, Linz, Austria, JKU Business School, Johannes Kepler University Linz, Austria
3Institute for Transport and Logistics Management, Vienna University of Economics and Business, Vienna, Austria

Abstract

In this work, we present the first outer approximation algorithm for multiobjective mixed-integer linear programming (MOMILP). The algorithm produces the non-dominated extreme points for MOMILP, and the facets of the convex hull of these points. Our algorithm uses an oracle which consists of solving single-objective weighted-sum problems. We show that analogous to existing outer approximation algorithms for multiobjective linear programming, our algorithm needs a number of oracle calls, which is polynomial in the number of facets of the convex hull of the non-dominated extreme points. As a consequence, for problems for which the weighted-sum problem is solvable in polynomial time, the facets can be computed in incremental polynomial time.

From a practical perspective, due to its outer approximation nature, the algorithm provides valid lower bound sets for use in multiobjective branch-and-bound algorithms at any stage of its execution and thus allows for early termination of such a bound computation. Moreover, the oracle produces Pareto optimal solutions, which makes the algorithm also attractive from the primal side. Finally, the oracle can also be called with any relaxation of the original MOMILP, and the obtained points and facets still provide a valid lower bound set. A computational study is done to assess the efficiency of our algorithm.

1 Introduction

Many if not all real-world problems involve several, often conflicting, objectives. Prominent examples are profitability or cost versus environmental concerns [20, 33] or customer satisfaction [13]. Whenever it is not possible to aggregate these conflicting objectives in a meaningful way, it is recommendable to produce a set of trade-off solutions to elucidate their trade-off relationship. One of the main optimality concepts used in this setting is Pareto optimality. A solution is Pareto optimal if no objective function value can be improved without deteriorating another. The associated points in the objective space (i.e., the space of objective function values) are called non-dominated. The usual goal is to find one Pareto optimal solution for each point on the non-dominated frontier, which is the set of all non-dominated points. For more background on multiobjective optimization, we refer to e.g., [13, 21].

In this work, we focus on multiobjective mixed-integer linear programming (MOMILP). General-purpose exact solution approaches for MOMILP can be divided into two classes: i) those that work
in the objective space (referred to as criterion or objective space search methods) and ii) those that work in the space of feasible solutions, which are mainly generalizations of branch-and-bound (B&B) algorithms (see, e.g., 1, 5, 24, 29, 30, 35). We note that the research is mainly focused on biobjective and combinatorial or integer problems, and extension to more objectives and mixed-integer variables, in particular from a computational point of view, is still in its infancy.

Generic methods which can handle more than two objectives have so far been restricted to objective space search methods, see, e.g., 11, 12, 36, 37 and the references therein. Most of these methods focus on the pure integer programming case, and not MOMILP. Objective space search methods solve a succession of single-objective problems in order to compute the set of Pareto optimal solutions. The main ingredient that can be worked on is the order in which the efficient solutions are generated and how many calls to the single objective solver are necessary [18, 19]. Research on multiobjective B&B algorithms provides many more possibilities. Increased efficiency can be obtained by employing different bounding procedures, tree exploration strategies, branching rules or pruning techniques, or by developing new ones [30]. Recent research shows the advantages of using lower bound sets [22] in the context of biobjective B&B [24, 29]. Lower bound sets are sets in the objective space, which contain the non-dominated frontier, and are a natural multiobjective extension of the lower bounds obtained by e.g., relaxations in single-objective optimization.

Our proposed outer approximation (OA) algorithm for MOMILP produces the non-dominated extreme points for the MOMILP, and the facets of the convex hull of these points (for more details, see Section 2). This is an interesting problem on its own and is also known as the case of parametric optimization, where we have the parameterization in the objective function. In this problem setting, the non-dominated extreme points are called breakpoints. So far, only inner approximation (IA) algorithms exist to solve this problem for MOMILP [28, 31, 32] (in contrast to this, for multiobjective linear programming (MOLP), there exist OA algorithms [6, 25, 27, 34] and IA algorithms [17]). One key advantage of using an OA algorithm is that by definition, at each iteration, a superset of the non-dominated frontier is available; thus, at each step of the algorithm, we have a valid lower bound set, which is useful in the context of multiobjective B&B. This means that early interruption is possible, e.g., as soon as it becomes clear that the B&B node can be fathomed. We note that current biobjective B&B algorithms use IA algorithms for computation of the lower bound set. As these approaches build the lower bounds set "from the inside", they only provide a valid lower bound set at termination, therefore interrupting them early does not yield a valid bound set. Moreover, while for biobjective ILPs the IA algorithm reduces to the well-known weighted-sum method/dichotomic search [2], the (computationally efficient) extension of IA algorithms for MOMILP with more than two objectives is not straightforward, see, e.g., [28, 31, 32].

Detailed contribution and outline. In this work, we present the first OA algorithm for MOMILP. Our algorithm is motivated by the OA approach of Benson [6] for MOLP. Our algorithm uses an oracle which consists of solving single-objective weighted-sum problems. We show that our algorithm needs a number of oracle calls which is polynomial in the number of facets of the convex hull of the non-dominated extreme points. As a consequence, if the weighted-sum problems are solvable in polynomial time, the facets can be computed in incremental polynomial time. These results extend and complement the results of [10], which showed that with an IA algorithm, the extreme points can be found in incremental polynomial time. From a practical perspective, next to providing a lower bound set for MOMILP at any point of its execution, each oracle call can also produce a new Pareto optimal solution. This makes its use within multiobjective B&B attractive also from a primal side. Finally, the algorithm also provides a valid lower bound set when the oracle is called with any relaxation of the original MOMILP, such as its LP-relaxation, or its LP-relaxation augmented with valid inequalities. This opens perspectives for multiobjective branch and cut algorithms.

In Section 2 we provide notations, definitions and other preliminaries needed in the remainder of the paper. Section 3 starts with a general outline of OA algorithms for multiobjective optimization. To work for a concrete problem class such as MOLP or MOMILP, a point separation oracles specific for the problem class is needed. We propose two separation oracles suited for MOMILP. The theoretical
runtime of the algorithm when using the presented oracles is discussed in Section 4. Section 5 contains a computational study to assess the efficiency of our algorithm and also discusses implementation details.

2 Notation, definitions and preliminaries

Let \( A \in \mathbb{Q}^{m \times n} \), \( C \in \mathbb{Q}^{p \times n} \), \( b \in \mathbb{Q}^m \) and \( n = n_1 + n_2 \). The feasible region of a MOMILP instance in decision space \( \mathbb{R}^n \) is defined as \( \mathcal{A} = \{ x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2} : Ax \geq b \} \). Throughout the paper, assume \( \mathcal{A} \) is bounded. The feasible points in objective space \( \mathbb{R}^p \) are \( \mathcal{Q} = \{ y : y = Cx, x \in \mathcal{A} \} \). This means matrix \( C \) projects feasible solutions from decision space to objective space. The multiobjective mixed-integer linear programming problem (using Pareto optimality as optimality concept) can be formulated as \( \min \{ y : y \in \mathcal{Q} \} \), where \( \min \) denotes the minimal elements of the set w.r.t. the coordinate-wise less-or-equal order. A point \( \hat{y} \in \mathcal{Q} \) is non-dominated, i.e. no \( y \leq \hat{y}, y \neq \hat{y} \) is in \( \mathcal{Q} \). It is a supported non-dominated point, i.e. there is a \( \hat{w} \in \mathbb{R}^p \) with \( \hat{w} > 0 \), such that it is the solution to the weighted-sum problem or weighted-sum scalarization (WSUM): \( \min \{ \hat{w}^T y : y \in \mathcal{Q} \} \). Note that compared to MOLP, in MOMILP there can be non-supported non-dominated points. The existence of such points justifies the need for multiobjective B&B algorithms, with which they can be found.

The set of supported non-dominated points can also be described by the intersection of \( \mathcal{Q} \) with the boundary of the polyhedron \( \mathcal{Q}^+ := \text{conv} \mathcal{Q} + \mathbb{R}_+^p \). The extreme points of this polyhedron are called non-dominated extreme points and are always elements of \( \mathcal{Q} \). In our problem setting, we are interested in the facets of \( \mathcal{Q}^+ \). The polyhedron is illustrated in Figure 1 and the non-dominated extreme points are the red points with the black circles around them.

If we expand \( \mathcal{Q} \) in the above definition, we get \( \mathcal{Q}^+ := \{ y : y = Cx, x \in \text{conv} \mathcal{A} \} + \mathbb{R}_+^p \). This, computing the extreme points or facets of \( \mathcal{Q}^+ \) is, in fact, a MOLP problem, as we are optimizing several objective functions over a polyhedron. With the caveat that the polyhedron is not in the correct input format since it is described by the convex hull of the feasible set of an MOMILP. However, it justifies the investigation of these MOLPs and we will show that it is not necessary to actually know the inequality representation of the polyhedron.

Analogously to MOMILP, the feasible set of an MOLP instance is the polyhedron \( \mathcal{P} := \{ x \in \mathbb{R}^n : Ax \geq b \} \), the feasible set in objective space is thus \( \mathcal{Q} := \{ y \in \mathbb{R}^p : y = Cx, x \in \mathcal{P} \} \) and we define \( \mathcal{Q}^+ \) in the same way. In MOLP, we are usually interested in a vertex and/or facet description of the set \( \mathcal{Q}^+ \).  

3 The outer approximation algorithm

In this section, we first give a general outline of OA algorithms for multiobjective optimization following the terminology introduced in [17]. The key ingredient in implementing such algorithms for concrete
Algorithm 1: Generic OA algorithm for multiobjective optimization

Data: initial approximation $S$ specified by double description

Result: $Q^+$ specified by double description

1 $S_0 = S$;
2 $insideVertices \leftarrow \emptyset$;
3 $i \leftarrow 0$;
4 while $\exists y^* \in vertices(S_i) : y^* \notin insideVertices$ do
5 \hspace{1em} $(status, H) \leftarrow$ separationOracle($y^*, S_i, Q^+$);
6 \hspace{1em} if status = inside then
7 \hspace{2em} $insideVertices \leftarrow insideVertices \cup \{y^*\}$;
8 \hspace{1em} else
9 \hspace{2em} $S_{i+1} \leftarrow S_i \cap H$;
10 \hspace{2em} $i \leftarrow i + 1$;
11 \hspace{1em} end
12 end
13 $Q^+ \leftarrow S_i$;

problem classes like MOLP or MOMILP is the specification and implementation of a point separating oracle for the respective problem class.

Definition 1 (Point separating oracle [17]). A point separating oracle for a polytope $Q^+ \subset \mathbb{R}^p$ is a black box algorithm which takes as input a point $y^* \in \mathbb{R}^p$ and returns as tuple $(status, H)$. The output is as follows: i) $(inside, \emptyset)$, if $y^* \in Q^+$, or ii) $(outside, H)$, where $H$ is a supporting hyperplane $H = \{y \in \mathbb{R}^p : w^Ty = \alpha\}$ of $Q^+$ such that $w^Ty^* < \alpha$ and $w^T\hat{y} \geq \alpha$ for each $\hat{y} \in Q^+$ (i.e., $H$ separates $y^*$ from $Q^+$).

The OA algorithm is described in Algorithm 1. The approximations $S = S_0 \supset S_1 \supset \ldots$ used within the algorithm are considered to be stored in a double description format, i.e., as vertices and facets. The algorithm starts with an initial approximation $S$. This initial approximation consists of the ideal point $y^I \in \mathbb{Q}_p$, whose $i$th component is the minimum value in the $i$th objective, together with the non-negative orthant, i.e., $S := y^I + \mathbb{R}_+^p$. It proceeds in an iterative fashion by checking the vertices of the current approximation $S_i$ for containment in $Q^+$ using the separation oracle. This is done repeatedly until for some $S_i$ the oracle answers with status $inside$ for each vertex, which means $S_i = Q^+$. The convergence-behaviour of the algorithm depends on the separation oracle. In the original work by Benson [6], the supporting hyperplanes to $Q^+$ were only face supporting. The first facet supporting hyperplane producing oracles were given in [3, 17] for MOLP. Hence, we are interested in facet supporting hyperplanes and we show in the following sections that our proposed separation oracle produces facets of $Q^+$.

Remark 1. It is easy to see, that at any point of the algorithm, we have $Q^+ \subseteq S_i$. Moreover, the algorithm still gives an OA of $Q^+$ when the separation oracle does not separate with respect to $Q^+$ but to any superset $S(Q^+) \supset Q^+$. This means that $Q^+$ can be replaced with any relaxation of the polyhedron $Q^+$ in the separation oracle when the goal is to obtain lower bound sets for multiobjective Branch and Bound algorithms.

Revisiting a separation oracle for MOLP. Let us revisit the original separation oracle for MOLPs
of the form \( \min \{ Cx : Ax \geq b, x \in \mathbb{R}^n \} \) (see, e.g., \([8, 17, 23]\)):

\[
\text{(Sep-Orig)} \quad \max b^T u - (y^*)^T w \\
\text{s.t. } A^T u - C^T w = 0 \\
\sum_{i=1,...,p} w_i = 1 \\
w_i \geq 0 \quad i = 1, \ldots, p \\
w_i \geq 0 \quad i = 1, \ldots, m
\]  

The point separation algorithm then works as follows: Given a feasible MOLP instance and a point \( y^* \in \mathbb{Q}^p \), we solve this LP. Let \( \alpha \) be the optimal value and \((\hat{w}, \hat{u})\) be an optimal solution. If \( \alpha > 0 \), we can use the \( w \) variables to construct a separating hyperplane \( \{ x \in \mathbb{R}^p : \hat{w}^T x = \alpha \} \). If \( \alpha = 0 \), then we return inside. We also observe that \( \hat{u} \) then is an optimal solution to the dual of the weighted-sum LP \( D(\hat{w}) : \max \{ b^T u : A^T u = \hat{w}^T C \} \) with respect to the weight \( \hat{w} \in \mathbb{Q}_+^p \).

When considering discrete multiobjective problems, let alone MOMILP, this point separating oracle has the flaw that a dual description analogous to linear programming is usually not available. Also just replacing dual feasible solutions in the above LP by primal feasible solution does not mitigate this problem, since the added \( w \) variables break the structure of the original problem.

With IA algorithms, there is a similar problem, as there we need to find a plane separating oracle. While the problem for IA algorithms is solved \([10]\) it is open for OA algorithms. In the following, we present two new point separating oracles that work for MOMILP and only interact with the problem instance at hand by solving weighted-sum scalarizations.

**A first separation oracle for MOMILP.** Let \( w_i \) be the coefficient of objective \( i, i = 1, \ldots, p \) in the desired hyperplane \( H \) and \( \alpha \) be the right-hand-side of \( H \).

\[
\text{(Sep-}y^*\text{)} \quad \min (y^*)^T w - \alpha \\
\text{s.t. } y^T w - \alpha \geq 0 \quad \forall y \in \mathbb{Q} \\
\sum_{i=1,\ldots,p} w_i = 1 \\
w_i \geq 0 \quad i = 1, \ldots, p
\]

Problem (Sep-\( y^*\)) encodes that \((w, \alpha)\) should induce a separating hyperplane by enforcing that all \( y \in \mathbb{Q} \) should be on the positive side of the hyperplane using constraints \([\text{FEAS}]\). Constraint \([\text{WSUM1}]\) is a normalisation constraint for the coefficients of the obtained hyperplane. It will be used in the proofs later on. If the objective function value of (Sep-\( y^*\)) is negative for a given \( y^* \), we get that \( y^* \notin \mathbb{Q}^+ \) and \((w, \alpha)\) gives the coefficients of the corresponding separating hyperplane. To deal with the infinite size of \( \mathbb{Q} \), we propose to solve (Sep-\( y^*\)) using a cutting-plane approach, where constraints \([\text{FEAS}]\) get separated: Given a solution \((\hat{w}, \hat{\alpha})\), there exists a violated constraint \([\text{FEAS}]\), if \( \min \{ \hat{w}^T y : y \in \mathbb{Q} \} < \hat{\alpha} \). Thus, the separation problem for \([\text{FEAS}]\) consists of solving a weighted-sum problem. More implementation details are given in Section \([5]\). We note that the separation oracle shares some similarity with the LPs used in local cut separation for single-objective mixed-integer linear programming \([3, 16]\).

We now prove that this is an implementation of a point separating oracle for MOMILP. To this end, we use that \( \mathbb{Q}^+ \) for the given MOMILP instance can be described as an MOLP instance as discussed in Section \([2]\). But since we only need to solve weighted-sum scalarizations of this MOLP in the separation steps of the cutting-plane approach above, we do not actually need the MOLP formulation itself, since this is equivalent to solving the weighted-sum scalarization of the original MOMILP. So we prove that given a feasible MOLP instance and a \( y^* \in \mathbb{R}_+^p \setminus \mathbb{Q}^+ \)—as in the OA algorithm—, if we consider the \( w \) variables of an optimal solution to problem (Sep-\( y^*\)) then there is an optimal solution to (Sep-Orig) with the same \( w \)-values. Thus, (Sep-\( y^*\)) can be used as a point separating oracle in the same way as (Sep-Orig).
To this end, we define \( F^O \) to be the optimal face to (Sep-Orig) and \( F^N \) the optimal face to (Sep-\( y^* \)). First, we prove that in (Sep-\( y^* \)) the \( \alpha \) variable is redundant in optimal solutions, although it provides us with useful information in the algorithm. Afterwards, we prove the aforementioned relationship between \( F^O \) and \( F^N \).

**Lemma 1.** For an optimal solution \((\hat{w}, \hat{\alpha}) \in F^N\) to problem (Sep-\( y^* \)), we have \( \hat{\alpha} = \min \{ \hat{w}^T y : y \in Q \} = \min \{ \hat{w}^T x : x \in P \} \).

**Proof.** Using constraints \([\text{FEAS}]\), we have \( \forall y \in Q : \alpha \leq \hat{w}^T y \). Since \( Q = \{Cx : x \in P\} \), we can rewrite \([\text{FEAS}]\) as \( \forall x \in P : \alpha \leq \hat{w}^T C x \). As this is the only requirement on \( \alpha \) in (Sep-\( y^* \)), choosing \( \alpha \) in this way is also feasible. \( \square \)

**Proposition 1.** There is a linear surjection \( f : F^O \to F^N \) that is idempotent in \( w \). For every \( (w, \alpha) \in F^N \), we have \( f^{-1}(w, \alpha) = \{(w, u) : u \text{ optimal for } D(w)\} \).

**Proof.** Let \( f(w, u) \mapsto (w, -b^T u) \). Idempotency and linearity follows by definition. We first prove the surjectivity. Let thus \((\hat{w}, \hat{\alpha}) \in F^N\). We construct a feasible solution for (Sep-Orig) that maps to \((\hat{w}, \hat{\alpha})\) and then prove its optimality.

Let \( \hat{u} \) be an optimal solution to problem \( (D(\hat{w})) \). We see that \((\hat{w}, \hat{u}) \) is a feasible solution to (Sep-Orig) and due to Lemma 1 and LP duality, we have \( f(\hat{w}, \hat{u}) = (\hat{w}, \hat{\alpha}) \). Let us assume that \((\hat{w}, \hat{u}) \) is not optimal, so there is a feasible \((w', u')\) with \( b^T u' - (y^*)^T w' > b^T \hat{u} - (y^*)^T \hat{w} \). Then, let \( \alpha' := b^T u' \) and we have \((y^*)^T w' - \alpha' = (y^*)^T w' - b^T u' < (y^*)^T \hat{w} - b^T \hat{u} = (y^*)^T \hat{w} - \hat{\alpha} \). Using LP duality again, we see that \((w', \alpha')\) is feasible for (Sep-\( y^* \)) since for all \( x \in P : w'^T C x \geq b^T u' = \alpha' \), which contradicts optimality of \((\hat{w}, \hat{\alpha})\).

For the second claim, \( f^{-1}(w, \alpha) \supseteq \{(w, u) : u \text{ optimal for } D(w)\} \) follows, since in the above proof the optimal \( \hat{u} \) can be chosen as an arbitrary optimal solution to \((D(\hat{w}))\). The other inclusion follows from Lemma 1 and LP duality. \( \square \)

From idempotency and surjectivity of \( f \) the following corollary follows, where \( \text{proj}_w \) denotes the orthogonal projection on the \( w \) variables.

**Corollary 1.** \( \text{proj}_w (F^O) = \text{proj}_w (F^N) \).

To rephrase the above considerations in the context of the OA algorithm: Whenever we solve the problem (Sep-\( y^* \)) and use the given weight \( w \), there also is an optimal solution to (Sep-Orig) with the same weight. In this sense, both problems are equivalent and we can use (Sep-\( y^* \)) to implement a point separating oracle.

Csirmaz\cite{17} argues that in optimal solutions \((\hat{w}, \hat{u})\) to (Sep-Orig), the weights \( \hat{w} \) constitute a facet supporting hyperplane iff \( \hat{w} \) is extremal among the convex polyhedron of all feasible \( w \). Clearly, although \((\hat{w}, \hat{u})\) might be an optimal extreme point solution to (Sep-Orig), \( \hat{w} \) might not be an extreme point of the set of all optimal \( w \). Thus, Csirmaz suggests to find a lexicographic minimal \( w \) among the optimal ones to resolve this issue. However, in our point separating oracle (Sep-\( y^* \)), the set of feasible \( w \) corresponds directly to these \( w \) as we see in Corollary 1. Thus, any optimal extreme point solution to (Sep-\( y^* \)) constitutes a facet supporting hyperplane to \( Q^\top \).

**Theorem 1.** The above algorithm using (Sep-\( y^* \)) is a point separating oracle that provides us with facet supporting inequalities.

A target cut-like separation oracle for MOMILP. Another separation oracle can be defined as follows. This oracle follows the target-cut paradigm introduced in\cite{14}. In the description below, we assume that all points encountered in the separation process are \( \geq 1 \). This is without loss of generality, as we can always add a large constant to all points. In the presentation of this oracle we start from the
dual side and then proceed to the primal side.

\[
(D-T\text{Sep-}y^*) \quad \max \sum_{i : \hat{y}_i \in Q} \lambda_i \\
\text{s.t.} \quad \sum_{i : \hat{y}_i \in Q} \lambda_i \hat{y}_i \leq y^* \\
\lambda_i \geq 0 \quad i : \hat{y}_i \in Q
\]  

(D-T-FEAS)  

(TSUM)

Constraints (D-T-FEAS) impose that \( y^* \) must be equal to a point, or dominated by a point which can be obtained by a linear combination of the points in \( Q \). Constraints (D-T-FEAS) ensure that this linear combination only has positive coefficients. The key insight is that the combination is that \( y^* \) only belongs to \( Q^+ \) iff the objective of (D-T\text{Sep-}y^*) is one or larger, see [14] for more details.

\[
(T\text{Sep-}y^*) \quad \min (y^*)^T w \\
\text{s.t.} \quad y^T w \geq 1 \\
\quad \quad \quad i = 1, \ldots, p
\]  

(TFEAS)  

(TWSUM)

Using the above ideas, we can implement a point separating oracle as follows. If an optimal solution \( \hat{w} \) of \((T\text{Sep-}y^*)\) has an objective value smaller than 1, \( \hat{w}^T y \geq 1 \) gives a separating hyperplane. Otherwise, we return inside. Similar to constraints (FEAS) of (Sep-\( y^* \)), we propose to solve \((T\text{Sep-}y^*)\) using a cutting-plane approach, where constraints (TFEAS) are separated by solving weighted-sum problems.

The reason this algorithm is a point separating oracle is as follows: We see that for any feasible solution \( \hat{w} \) with objective value \( (y^*)^T \hat{w} < 1 \), the hyperplane \( \{ x \in \mathbb{R}^p : w^T x = 1 \} \) again separates \( y^* \) from \( Q \), as long as \( Q \subseteq \mathbb{R}_+^p \). Moreover, in case the minimum value is at least 1, then there is no separating hyperplane, hence \( y^* \in Q^+ \). Albeit, this hyperplane is not necessarily supporting \( Q \). However, using the property again that there is an MOLP instance for a given MOMILP as discussed in Section 2, we can perceive problem \((T\text{Sep-}y^*)\) again as an LP. Then we observe that the feasible set of \((T\text{Sep-}y^*)\) is polar dual to \( Q^+ \). Thus, every extreme point solution to \((T\text{Sep-}y^*)\) provides us with a facet of \( Q^+ \).

**Theorem 2.** The above algorithm using \((T\text{Sep-}y^*)\) is a point separating oracle that provides us with facet supporting inequalities.

### 4 Comments on theoretical running time

Böckler [8] conducts a comprehensive theoretical running time study of the general IA and OA algorithms for MOLP using the framework of output-sensitive running-time analysis (cf. also [9]). Therein it is proven that given a point separating oracle that produces facets only, the general OA algorithm has polynomial running-time in the input size and the number of facets of \( Q^+ \), for each fixed number of objectives. Moreover, it is shown that it also runs in incremental polynomial time considering the number of facets. This means that the time it takes the algorithm to compute the next facet given \( k \in \mathbb{N} \) already computed facets is in \( O(\frac{k}{k^2} (T_O + k \log k)) \), where \( T_O \) is the running time of the oracle.

**Corollary 2.** The OA algorithm with the point separating oracles from Section 3 runs in incremental-polynomial time considering the facets (or facets and extreme points) as output.

Conversely, it is shown that for the general IA algorithm, if the analog plane separating oracle produces extreme points only, the algorithm runs in incremental polynomial time considering the number of nondominated extreme points.

In the case of the IA algorithm, Böckler and Mutzel [10] prove that it is only necessary to solve weighted-sum oracles instead of LP oracles with added constraints, which makes the IA algorithms very...
useful for multiobjective combinatorial problems where additional constraints would destroy the combinatorial structure of the feasible set. Moreover, they prove that if the weighted-sum problem of a given multiobjective combinatorial optimization problem can be solved in polynomial time, then the extreme points can be found in incremental polynomial time for every fixed number of objectives.

Since the OA algorithm with our new point separating oracles only interacts with the underlying MOMILP problem by means of weighted-sum problems, we have a similar theorem for the computation of facets.

**Theorem 3.** If the weighted-sum problem for a given MOMILP is polynomial time solvable, then the facets of $Q^+$ can be computed in incremental polynomial time.

## 5 Computational results

We have implemented the proposed OA algorithms using CPLEX 12.9 as LP/MILP-solver and the parma polyhedral library (ppl) [4] for vertex enumeration, implementation details are given further down.

**Instances.** We use *multiobjective assignment problem (MAP)* instances with three objectives and *multiobjective knapsack problem (MKP)* instances with three, four and five objective from the literature [26] for our tests. Similar instances were also used in experiments for IA algorithms for MOMILP [28, 31, 32].

The MAP instances are denoted as set MAP and are constructed as follows: The number of agents and tasks is the same, and ranges from 5 to 50 in increments of 5. The objective function coefficients are random integers in the range $[1, 20]$. There are 100 instances in the set. The MKP instances are denoted as set MKP and are constructed as follows: Both the profit and the weights are random integers from the interval $[1, 1000]$. The budget is calculated as half of the sum of the weight of all items, rounded up to the next integer. There are 160 instances in the set, 100 instances are with three objectives (10 to 100 items), 40 with four (10 to 40 items), and 20 with five (10 to 20 items). To fit into our minimization setting, we transform the MKP into a minimization version with negative objective coefficients. When using (TSep-$y^*$) as oracle, we add the negative sum of all objective coefficients to each coordinate so that the obtained points are ensured to be positive.

**Implementation details.** The focus in the implementation of our algorithms was to find a good balance between numerical accuracy and speed. We note that IA algorithms can suffer from numerical issues, for example in [32] different versions of the same IA algorithm are tested, and some versions cannot find the complete set of non-dominated extremes points for some instances. In OA, numerical issues can lead to missing to cut-off points, which do not belong to $Q^+$. However, even if we miss to cut-off some of these points, the obtained solution is still an outer approximation. To check the accuracy of our outer approximation, we compared the obtained extreme points for the MAP (after rounding them to the nearest integer) with the extreme points obtained by bensolve [27], which is a MOLP solver (as the assignment problem is totally unimodular, it can be solved as LP). For all instances, the obtained points coincide.

For the initial approximation $S$, the ideal point $y^I$ can be obtained by solving $\min_{y \in Q} e_i^T y$ for unit-vector $e_i$. Moreover, we initialize the separation LPs by adding constraints $\{\text{FEAS}\}$, resp., $\{\text{TFEAS}\}$ induced by the solutions obtained for the $p$ problems used for calculating the ideal point. When separating constraints $\{\text{FEAS}\}$, resp., $\{\text{TFEAS}\}$, we use a tolerance of $\varepsilon = 1e-3$ for checking violation. Once a constraint $\{\text{FEAS}\}$, resp., $\{\text{TFEAS}\}$ is added to the separation LP, we leave it there for the remainder of the algorithm (i.e., for all subsequent separation oracle calls). In the separation LPs, we turn on the numerical emphasis parameter of CPLEX, and set the feasibility and optimality tolerances to $1e-9$, i.e., the most accurate value possible. For checking the result of the separation oracle (i.e., the objective value of the separation LP against zero, resp., one), we also use a tolerance of $\varepsilon = 1e-3$. When solving MKP with (TSep-$y^*$), we set the right-hand-side of $\{\text{TFEAS}\}$ to the sum of the objective coefficients instead of one, as this lead to increased numerical stability.
We use ppl with integer numbers as input, as using fractional numbers leads to (more) numerical instabilities. In order to do so, we scale each \((w, \alpha)\) obtained from the separation oracle by \(10^9\) and take the integer part of the obtained number. For checking if a point \(y^* \in \text{vertices}(S_i)\) is in \(Q^+\), we exploit that all our instances have integer coefficients, thus all points in \(Q\) (and consequently, the extreme points of \(Q^+\)) are integer. When a point \(y^*\) is within 1e-3 to a point in \(y' \in \text{insideVertices}\) for each coordinate, we consider \(y^* = y'\) and do not call the separation oracle for it. For each \(S_i\), we check the points in the order as they are made available by ppl. Moreover, for updating the outer approximations, we use a slightly different strategy compared to the outline in Algorithm 1. We do not immediately calculate \(S_i \cap H\) and move to \(S_{i+1}\) once we discovered a \(y^* \notin \text{insideVertices}\), but check all \(y^* \in \text{vertices}(S_i)\), collect the obtained separating hyperplanes, and then add them all at the same time to obtain the next outer approximation. This approach turned out to be faster and more numerically stable in preliminary computations. Finally, we also add every point found when separating (FEAS), resp., (TFeas) to \(\text{insideVertices}\), as these points are obtained from the solution of a weighted-sum problem, and thus are supported non-dominated points.

**Results.** In Tables 1 and 2 we present results for runs with timelimits of 10 seconds, 100 seconds, and 600 seconds. The results are aggregated by size (for MAP), resp., number of items and objectives (for MKP), the number of aggregated instances is always ten. We report the average number of obtained facets (\#fac), and the number of instances solved out of ten (\#sol). In our case, solving an instance means that all extreme points and facets have been computed and the algorithm terminated.

For the MAP, the performance of both separation oracles is quite similar, within 600 seconds, all instances except the ones with size 50 can be solved, and within 100 seconds all up to size 30 can be solved. For the MKP, all instances can be solved within 100 seconds. TSep-\(y^*\) seems a little more effective, only 11 out of 160 instances are not solved within 10 seconds when using this separation oracle. The number of obtained facets for both separation oracles is always quite similar. Note that due to numerical instabilities, these numbers can differ slightly even if both oracles manage to solve an instance.

| size | 10s | 100s | 600s | 10s | 100s | 600s |
|------|-----|------|------|-----|------|------|
| #fac | #sol | #fac | #sol | #fac | #sol | #fac | #sol |
| 5    | 13.0 | 10   | 13.0 | 10  | 13.0 | 10   | 13.4 | 10   |
| 10   | 76.6 | 10   | 76.6 | 10  | 76.6 | 10   | 77.8 | 10   |
| 15   | 164.4| 10   | 164.4| 10  | 164.4| 10   | 166.6| 10   |
| 20   | 267.7| 3    | 309.8| 10  | 309.8| 10   | 268.0| 4    |
| 25   | 117.0| 0    | 498.7| 10  | 498.7| 10   | 117.3| 0    |
| 30   | 54.3 | 0    | 731.0| 10  | 731.0| 10   | 53.2 | 0    |
| 35   | 35.2 | 0    | 671.1| 0   | 968.2| 10   | 34.3 | 0    |
| 40   | 26.7 | 0    | 399.8| 0   | 1334.7| 10  | 26.0 | 0    |
| 45   | 20.9 | 0    | 262.2| 0   | 1604.7| 10  | 19.6 | 0    |
| 50   | 15.7 | 0    | 176.8| 0   | 1514.3| 0   | 15.6 | 0    |

**6 Conclusion**

We present the first outer-approximation algorithm for MOMILP and prove its theoretical running-time properties. With the practical application in multiobjective B&B algorithms in mind, our experiments show that lower bound sets can be computed quickly.

Future work should investigate the efficacy of this approach in practical B&B algorithms. Also extensions to general non-linear optimization seem possible and should be investigated.
Table 2: Results for instances MKP with $p = 3, 4, 5$ aggregated by number of items

| items | 10s | 100s | 600s | 10s | 100s | 600s |
|-------|-----|------|------|-----|------|------|
|       | #fac | #sol | #fac | #sol | #fac | #sol |
| 10    | 8.1  | 10   | 8.1  | 10   | 8.1  | 10   |
| 20    | 27.2 | 10   | 27.2 | 10   | 26.9 | 10   |
| 30    | 53.4 | 10   | 53.4 | 10   | 53.6 | 10   |
| 40    | 74.6 | 10   | 74.6 | 10   | 75.0 | 10   |
| 50    | 102.8| 10   | 102.8| 10   | 103.1| 10   |
| 60    | 147.9| 10   | 147.9| 10   | 147.9| 10   |
| 70    | 198.4| 9    | 198.4| 10   | 200.2| 10   |
| 80    | 248.2| 8    | 249.4| 10   | 246.7| 10   |
| 90    | 280.1| 3    | 308.2| 10   | 289.5| 6    |
| 100   | 277.9| 0    | 385.9| 10   | 288.6| 3    |

| items | 10s | 100s | 600s | 10s | 100s | 600s |
|-------|-----|------|------|-----|------|------|
|       | #fac | #sol | #fac | #sol | #fac | #sol |
| 10    | 7.4  | 10   | 7.4  | 10   | 7.5  | 10   |
| 20    | 26.5 | 10   | 26.5 | 10   | 26.9 | 10   |
| 30    | 49.9 | 10   | 49.9 | 10   | 48.8 | 10   |
| 40    | 99.8 | 10   | 99.8 | 10   | 99.0 | 10   |

| items | 10s | 100s | 600s | 10s | 100s | 600s |
|-------|-----|------|------|-----|------|------|
|       | #fac | #sol | #fac | #sol | #fac | #sol |
| 10    | 7.6  | 10   | 7.6  | 10   | 7.4  | 10   |
| 20    | 24.0 | 10   | 24.0 | 10   | 23.7 | 10   |

Acknowledgements. This work is supported by the Austrian Science Fund (FWF), project P 31366-NBL.

References

[1] Nathan Adelgren and Akshay Gupte. Branch-and-bound for biobjective mixed integer programming. *arXiv preprint arXiv:1709.03668*, 2017.

[2] Yash P Aneja and Kunhiraman PK Nair. Bicriteria transportation problem. *Management Science*, 25(1):73–78, 1979.

[3] David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. Tsp cuts which do not conform to the template paradigm. In *Computational combinatorial optimization*, pages 261–303. Springer, 2001.

[4] Roberto Bagnara, Patricia M Hill, and Enea Zaffanella. The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1-2):3–21, 2008.

[5] Pietro Belotti, Banu Soylu, and Margaret M Wieck. A branch-and-bound algorithm for biobjective mixed-integer programs. *Optimization Online*, 2013.

[6] Harold P Benson. An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem. *Journal of Global Optimization*, 13(1):1–24, 1998.

[7] Harold P Benson and Erjiang Sun. A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program. *European Journal of Operational Research*, 139(1):26–41, 2002.
[8] F. Bökl er. Output-Sensitive Complexity for Multiobjective Combinatorial Optimization with an Application to the Multiobjective Shortest Path Problem. PhD thesis, TU Dortmund University, 2018.

[9] F. Bökl er, M. Ehrgott, C. Morris, and P. Mutzel. Output-sensitive complexity for multiobjective combinatorial optimization. Journal of Multi-Criteria Decision Analysis, 24(1–2):25–36, 2017.

[10] Fritz Bökl er and Petra Mutzel. Output-sensitive algorithms for enumerating the extreme nondominated points of multiobjective combinatorial optimization problems. In Algorithms-ESA 2015, pages 288–299. Springer, 2015.

[11] Natasha Boland, Hadi Charkhgard, and Martin Savelsbergh. The l-shape search method for triobjective integer programming. Mathematical Programming Computation, 8(2):217–251, 2016.

[12] Natasha Boland, Hadi Charkhgard, and Martin Savelsbergh. The quadrant shrinking method: A simple and efficient algorithm for solving tri-objective integer programs. European Journal of Operational Research, 260(3):873–885, 2017.

[13] Kris Braekers, Richard F Hartl, Sophie N Parragh, and Fabien Tricoire. A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience. European Journal of Operational Research, 248:428–443, 2016.

[14] Christoph Buchheim, Frauke Liers, and Marcus Oswald. Local cuts revisited. Operations Research Letters, 36(4):430–433, 2008.

[15] Altannar Chinchuluun and Panos M Pardalos. A survey of recent developments in multiobjective optimization. Annals of Operations Research, 154(1):29–50, 2007.

[16] Vašek Chvátal, William Cook, and Daniel Espinoza. Local cuts for mixed-integer programming. Mathematical Programming Computation, 5(2):171–200, 2013.

[17] Laszlo Csirmaz. Inner approximation algorithm for solving linear multiobjective optimization problems. Optimization, pages 1–25, 2020.

[18] Kerstin Dächert and Kathrin Klamroth. A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. Journal of Global Optimization, 61(4):643–676, 2015.

[19] Kerstin Dächert, Kathrin Klamroth, Renaud Lacour, and Daniel Vanderpooten. Efficient computation of the search region in multi-objective optimization. European Journal of Operational Research, 260(3):841–855, 2017.

[20] Emrah Demir, Tolga Bektas, and Gilbert Laporte. The bi-objective pollution-routing problem. European Journal of Operational Research, 232(3):464–478, 2014.

[21] Matthias Ehrgott. Multicriteria optimization, volume 491. Springer Science & Business Media, 2005.

[22] Matthias Ehrgott and Xavier Gandibleux. Bound sets for biobjective combinatorial optimization problems. Computers & Operations Research, 34(9):2674–2694, 2007.

[23] Matthias Ehrgott, Andreas Löhne, and Lizhen Shao. A dual variant of benson’s ”outer approximation algorithm” for multiple objective linear programming. Journal of Global Optimization, 52(4):757–778, 2012.

[24] Sune Lauth Gadegaard, Lars Relund Nielsen, and Matthias Ehrgott. Bi-objective branch-and-cut algorithms based on lp relaxation and bound sets. INFORMS Journal on Computing, 31(4):790–804, 2019.
[25] Andreas H Hamel, Andreas Löhne, and Birgit Rudloff. Benson type algorithms for linear vector optimization and applications. *Journal of Global Optimization*, 59(4):811–836, 2014.

[26] G. Kirilik and S. Sayin. Multiobjective optimization library. URL: http://home.ku.edu.tr/~moolibrary/.

[27] Andreas Löhne and Benjamin Weißing. The vector linear program solver bensolve–notes on theoretical background. *European Journal of Operational Research*, 260(3):807–813, 2017.

[28] Özgür Özpeynirci and Murat Köksalan. An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. *Management Science*, 56(12):2302–2315, 2010.

[29] Sophie N Parragh and Fabien Tricoire. Branch-and-bound for bi-objective integer programming. *INFORMS Journal on Computing*, 31(4):805–822, 2019.

[30] Anthony Przybylski and Xavier Gandibleux. Multi-objective branch and bound. *European Journal of Operational Research*, 260(3):856–872, 2017.

[31] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. 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.

[32] Anthony Przybylski, Kathrin Klamroth, and Renaud Lacour. A simple and efficient dichotomic search algorithm for multi-objective mixed integer linear programs. *arXiv preprint arXiv:1911.08937*, 2019.

[33] Tânia Rodrigues Pereira Ramos, Maria Isabel Gomes, and Ana Paula Barbosa-Póvoa. Planning a sustainable reverse logistics system: Balancing costs with environmental and social concerns. *Omega*, 48:60–74, 2014.

[34] Lizhen Shao and Matthias Ehrgott. Approximating the nondominated set of an molp by approximately solving its dual problem. *Mathematical Methods of Operations Research*, 68(3):469–492, 2008.

[35] Thomas Stidsen, Kim Allan Andersen, and Bernd Dammann. A branch and bound algorithm for a class of biobjective mixed integer programs. *Management Science*, 60(4):1009–1032, 2014.

[36] Satya Tamby and Daniel Vanderpooten. Enumeration of the nondominated set of multiobjective discrete optimization problems. *INFORMS Journal on Computing*, 2020.

[37] Weihua Zhang and Marc Reimann. A simple augmented $\epsilon$-constraint method for multi-objective mathematical integer programming problems. *European Journal of Operational Research*, 234(1):15–24, 2014.