An Adaptive Patch Approximation Algorithm for bicriteria convex mixed-integer problems

Erik Diessel

Fraunhofer ITWM, Fraunhofer Platz 1, 67663 Kaiserslautern, Germany

ARTICLE HISTORY
Compiled 21st October 2020

Abstract
Pareto frontiers of bicriteria continuous convex problems can be efficiently computed and optimal theoretical performance bounds have been established. In the case of bicriteria mixed-integer problems the approximation of the Pareto frontier becomes however significantly harder.

In this paper, we propose a new algorithm for approximating the Pareto frontier of bicriteria mixed-integer programs with convex constraints. Such Pareto frontiers are composed of patches of solutions with shared assignments for the discrete variables. By adaptively creating such a patchwork, our algorithm is able to create approximations that converge quickly to the true Pareto frontier. As a quality measure we use the difference in hyper-volume between the approximation and the true Pareto frontier.

At least a certain number of patches is required to obtain an approximation with a given quality. This patch complexity gives a lower bound on the number of required computations. We show that our algorithm performs a number of optimization steps that is of a similar order as this lower bound. We provide an efficient MIP-based implementation of this algorithm.

The efficiency of our algorithm is illustrated with numerical results showing that our algorithm has a strong theoretical performance guarantee while being competitive with other state-of-the-art approaches in practice.

KEYWORDS
Bicriteria optimization, mixed-integer programming, approximation, rate of convergence

AMS CLASSIFICATION
90C29, 90C11, 90C59, 90-08

1. Motivation and Concepts

Optimization problems in practice naturally have multiple conflicting goals. To identify optimal trade-offs with respect to these goals, multicriteria optimization aims to compute the corresponding Pareto frontiers. For these applications, the ability to compute efficiently a good approximation to the Pareto frontier is needed. A large fraction of models for typical real-world decision problems are mixed-integer programs that integrate discrete with continuous decision variables. In industrial applications the number of continuous variables typically clearly dominates the number of discrete variables. Existing algorithms for bicriteria optimization of mixed-integer programs are able to compute approximations of the Pareto frontier ([1]) provides a survey, recent algorithms include [2,3]). Although for many of these algorithms it is known that their results converge to the true Pareto frontier, state-of-the-art methods do

Erik Diessel (Corresponding author) Email: erik.diessel@itwm.fraunhofer.de ORCiD: 0000-0002-0671-9169
not provide specific bounds on the convergence speed. Asymptotically optimal bounds for the number of optimization steps needed to compute the exact Pareto frontier are only known for algorithms on pure integer problems in the case of two \cite{5} and three criteria \cite{6}.

For the simpler case without discrete variables however, approximation algorithms for continuous convex bicriteria problems with strong guarantees on the convergence speed have been established \cite{7}. This bound of $O(1/n^2)$ on the Hausdorff distance after $n$ optimization steps is asymptotically optimal since any approximation with $n$ segments of a convex two-dimensional set has an error of $\Omega(1/n^2)$ in the worst-case \cite{8}. In this paper we give an analogous result for convex mixed-integer programs, which is however instance-specific. We present an algorithm that only needs a number of optimization steps of the same order as the number of parts necessary to represent an approximation of the obtained quality.

We consider bicriteria mixed-integer programs with constraints that are convex in the continuous variables, i.e. each constraint can be represented by an inequality $g(c,d) \leq 0$ where the function $g$ is convex in the continuous variables $c \in \mathbb{R}^k$ for each fixed choice of the discrete variables $d \in \mathbb{Z}^m$. A patch is a set of solutions for a mixed-integer program with shared assignments for the discrete variables. Interpolation can be performed between every two points on a patch since creating convex combinations maintains feasibility due to convexity of the constraints. Thus in objective space this corresponds to a segment in the case of linear objective functions. In Figure 1 some patches are shown in objective space.

As an approximation quality measure we use a measure based on the hypervolume \cite{9}. This difference volume measure is given by the volume of the region between the true Pareto frontier and the approximation. The use of this measures enables an adaptive choice on the number of parts used in different regions of the Pareto frontier. In areas where a single patch is already close to the true Pareto frontier no further patches are needed. This distinguishes our approach from classical considerations of the approximation quality where uniform grids on the nondominated set are required which leads to restrictions on the efficiency \cite{10}.

Our algorithm iteratively computes patches that are then added to the approximation. The computation of the new patch is done adaptively based on the current approximation to ensure a maximal improvement of the difference volume.

Our convergence analysis is done on a per-instance basis which allows for more fine-grained results than a global worst-case analysis. We show that the number of optimization steps required by our algorithm for a bicriteria programming instance is of a similar order as the number of patches required to represent an approximation to the corresponding Pareto frontier with the given quality. This deep theoretical analysis is combined with numerical evaluations on benchmarks that show that the practical efficiency of our algorithm is competitive with other state-of-the-art algorithms.

![Figure 1](image.png)

Figure 1. An illustration of three patches in objective space.
1.1. Related work

A large number of multicriteria optimization methods has been developed. However, the still large amount of new methods proposed in the recent years shows that current algorithmic techniques are not yet satisfactory in all application areas. We describe here a non-exhaustive subset of the methods in the literature to illustrate the main research directions.

Available Algorithms for computing Pareto frontiers of bicriteria mixed-integer programs (Bicriteria MIPs) can be classified into two types:

- Algorithms based on branch-and-bound that perform recursive branching on some variables [11–14].
- Algorithms based on scalarization that generate special scalarized MIPs that are then solved to optimality to obtain a new point and corresponding information about the Pareto frontier [3,4,15–18].

Only a small number of algorithms aim to compute approximations for which explicit quality guarantees were shown. For example, the work [19] provides an algorithm which is guaranteed to output $\varepsilon$-approximate Pareto frontiers. In most works (e.g. [18]) the approximation is only considered in an ad-hoc way, i.e. by specifying a stopping rule for an algorithm that is designed to compute the whole Pareto frontier exactly.

Previous approaches using patches. A limited number of works in multiciriteria optimization use the concept of a patch.

The notion of patches in multicriteria optimization is introduced in [20] along with applications to radiotherapy planning. Patches were used for performing a Pareto-navigation over the set of solutions of a multicriteria radiotherapy planning problem in [21].

In [13] the concept of a slice is presented which is similar to our concept of a patch. A bicriteria optimization problem is reduced to slice problems. For a fixed assignment to the discrete variables a corresponding feasible assignments to the continuous variables is computed. The assignment to the discrete variables is however fixed before, and not chosen adaptively as in our approach.

Patches are used implicitly in [11,22]. In this approach, after fixing the discrete variables, extreme Pareto points of the obtained continuous multicriteria problem are computed. The convex hull of these extreme points, which corresponds to a patch, is used for checking Pareto-efficiency. A correction to this check of Pareto-efficiency is provided in [23] along with some further improvements of the algorithm. In [18] a patch is implicitly computed in a line detection step, however only with fixed objective values.

1.2. Motivation and goals for our algorithm

Deficits of current approaches. All of the discussed algorithms for bicriteria mixed-integer programs have the following properties in common:

- The algorithms all compute the exact Pareto frontier (when given enough time).
- However, the algorithms are not adapted to the use case of obtaining just an approximation of the Pareto frontier: When stopping the algorithm earlier, no clear guarantee on the approximation quality is provided. Although some parameters often indirectly control the approximation quality, no analysis relating them to widely used quality measures is given.
- There is no comprehensive theoretical analysis of the running time of the algorithms as a function of the approximation quality. The implicit limits on the approximation quality are only considered in an ad-hoc way, i.e. by specifying a stopping rule for an algorithm that is designed to compute the whole Pareto frontier exactly.
quality imposed by the choice of parameters or early stopping is not analysed.

The characteristics of our approach. In this paper, we develop an algorithm which is designed contrarily to the above characteristics. We pose the following goals to achieve with our method:

- The approximation returned by the algorithm in each step should be an inner approximation, i.e. all points in the approximation should correspond to feasible solutions that can be directly computed.
- The algorithm should provide an approximation of the Pareto frontier with a clearly quantified approximation guarantee at every step. In particular, the algorithm can be stopped at every moment and we still obtain a result with a clear approximation guarantee, i.e. the algorithm should be an anytime algorithm [24]. The approximation quality obtained until each given time limit should be competitive with every other algorithm that requires a similar amount of time.
- The approximation quality obtained by the algorithm, respectively the needed number of iterations to reach some given accuracy, should be bounded with respect to some intrinsic property of the problem we solve. In particular, the performance of the algorithm should be not too far away from a theoretical optimum that is given by the number of parts minimally needed to describe an approximation to the Pareto frontier.

1.3. Results and Organization of this paper

In section 2 we describe the basics concepts of multicriteria optimization and approximation of Pareto frontiers. We describe the concept of a patch and introduce the difference volume as our approximation quality measure. This leads to the notion of patch complexity that measures the complexity for approximating the Pareto frontier inherent to a problem. We then introduce in section 3 the adaptive patch approximation algorithm and establish a corresponding convergence result based on the patch complexity. The proven upper bound on the number of iterations is of the same order as the lower bound given by the patch complexity. In the subsequent section 4 we describe how the iterations of the adaptive patch approximation algorithm can be realized. We provide a mixed-integer program formulation for finding an optimal patch with only about twice the number of variables and constraints of the original mixed-integer program. We show that it is sufficient to solve a constant number of these mixed-integer programs per iteration.

The developed algorithm is evaluated numerically in section 5 using several benchmark problems from the literature. The results show that the running time of our algorithm is competitive with other state-of-the-art approaches.

2. The complexity of approximating Pareto frontiers with patches

2.1. Bicriteria convex mixed-integer problems

Our algorithm works with arbitrary bicriteria convex mixed-integer problems. In the following we characterize this problem class and discuss the corresponding theoretical notions. For a comprehensive theoretical introduction to the topic of multicriteria optimization, see [25].

We present here the general case of $K \in \mathbb{N}, K \geq 2$ criteria although our algorithm will focus
later on the case $K = 2$. A general multicriteria program has the form

$$\begin{align*}
\text{optimize} & \quad f(x) \\
\text{s. t.} & \quad x \in X
\end{align*}$$

where $f$ is a vector-valued function $f : X \to \mathbb{R}^K$ with components $f(x) = (f_1(x), \ldots, f_K(x))^T$. We assume throughout this paper that all objectives should be minimized. Our algorithm treats the case of bicriteria mixed-integer convex programs, i.e. there are $K = 2$ criteria and the feasible set $X$ in decision space can be represented in the form

$$X = \left\{ (c, d) \in \mathbb{R}^k \times \mathbb{Z}^m \mid g_i(c, d) \leq 0 \text{ for each } i \in I \right\}$$

where the function $g_i$, $i \in I$ is for each fixed $d \in \mathbb{Z}^m$ convex w.r.t. $c \in \mathbb{R}^k$. We consider here the case that each of the component objective functions $f_i$, $i = 1, \ldots, n$ is an affine linear function. Note that when solving minimization tasks, this restriction to linear objective functions instead of general convex objective functions does not reduce the class of problems that we can model. A convex objective function can be replaced by a new variable and the objective function shifted into the constraints. This class of problems also contains the widely considered class of bicriteria mixed-integer linear programs.

An ordering cone is a cone $C \subseteq \mathbb{R}^K$ that is used to describe a dominance relation between two multicriteria objective vectors $z^{(1)} \in \mathbb{R}^n$ and $z^{(2)} \in \mathbb{R}^K$. The solution $z^{(1)}$ is said to dominate $z^{(2)}$ with respect to the ordering cone $C$ if $z^{(2)} \in z^{(1)} + (C \setminus \{0\})$.

Let $Z := f(X) \subseteq \mathbb{R}^K$ be the (nonempty) set of all feasible vectors in objective space. The goal of multicriteria optimization is to compute (representations of) the set of nondominated objective vectors $N(Z)$. With respect to the ordering cone $C$ this set is defined as

$$N(Z) := Z \setminus (Z + (C \setminus \{0\}))$$

In the following we assume for simplicity that the standard ordering cone $C = \mathbb{R}^K_\geq$ is used. In particular, this means that we want to minimize with respect to all objective functions.

The Nadir point of $Z$ is given by the vector $z^N(Z)$ with components

$$z^N_i(Z) := \sup\{z_i \mid z \in N(Z)\}, \quad i = 1, \ldots, n$$

representing the maximal trade-off that we can make in each component. The ideal point of $Z$ is given by the vector $z^I(Z)$ with components

$$z^I_i(Z) := \inf\{z_i \mid z \in Z\}, \quad i = 1, \ldots, n$$

representing the optimum that can be achieved in each objective function. Figure 2 illustrates a nondominated set with ideal and Nadir point.

### 2.2. Patches

The basis of our algorithm forms the concept of a patch. It makes use of the fact that the variables of a mixed-integer program can be decomposed into continuous and discrete parts. We can thus represent the set of feasible vectors $X$ in decision space as

$$X \subseteq \{(c, d) \mid c \in \mathbb{R}^k, d \in \mathbb{Z}^m\}.$$
A patch is a subset of the feasible vectors in $X$ that share an assignment to the discrete variables. A patch $p_d$ corresponding to the assignment $d \in \mathbb{Z}^m$ is given as a subset of the feasible vectors in decision space sharing this discrete assignment

$$p_d \subseteq \{(c, d) \in X \mid c \in \mathbb{R}^k\}.$$  

The points of a patch only differ by their assignments to the continuous variables. Patches are convex sets due to the convexity with respect to the continuous variables of the mixed-integer program. In particular, we can perform a linear interpolation between two solutions $x_1 = (c_1, d) \in p_d$ and $x_2 = (c_2, d) \in p_d$ of a single patch while still maintaining feasibility:

$$\lambda x_1 + (1 - \lambda) x_2 \in X \quad \text{for all} \ \lambda \in [0, 1].$$

Segment patches. By using this interpolation property, two points are sufficient to describe an infinite number of solutions. Given two feasible vectors $x_1 = (c_1, d) \in X, x_2 = (c_2, d) \in X$ with shared discrete assignment $d \in \mathbb{Z}^m$, we can generate a patch as the line segment connecting the two vectors. The corresponding segment patch $P^{x_1,x_2}$ is given by all convex combinations of the vectors, i.e.

$$P^{x_1,x_2} := \{\lambda x_1 + (1 - \lambda) x_2 \mid \lambda \in [0, 1]\}.$$  

By projecting this to the objective space, we obtain segments.

**Definition 2.1** (Segment patch in objective space). Let $(c_1, d) \in X$ and $(c_2, d) \in X$ be two (not necessarily distinct) feasible solutions with shared values for the discrete variables.

Let $z^{(1)} \in (f_1(c_1, d), \ldots, f_K(c_1, d)) + \mathbb{R}_{\geq 0}^K$ and $z^{(2)} \in (f_1(c_2, d), \ldots, f_K(c_2, d)) + \mathbb{R}_{\geq 0}^K$ be two points that are contained in the ordering cones of the corresponding objective vectors. Then, the line segment between $z^{(1)}$ and $z^{(2)}$ is a segment patch in objective space, or simply segment patch.

The simple representation of a segment patch by two vectors allows for an efficient search procedure by our algorithm. We use the symbol $s$ to denote such a segment patch. In the case of two criteria, we denote the components of the two endpoints $z^{(1)}$ and $z^{(2)}$ by $X_1(s) = z^{(1)}_1$, $X_2(s) = z^{(2)}_1$ and $Y_1(s) = z^{(1)}_2$, $Y_2(s) = z^{(2)}_2$. An illustration of a set of segment patches in...
objective space is given in Figure 3.

Figure 3. Illustration of two segment patches in a two-dimensional objective space. The patch \(s\) corresponds to the discrete assignment \(d\) and has endpoints given as \((X_1(s), Y_1(s))\) and \((X_2(s), Y_2(s))\). Another patch corresponds to a different discrete assignment \(d'\).

Scaling Assumption. In order to make the objective values dimensionless, we require the following scaling assumption which is also used in [12].

Assumption 2.1 (Scaling). The nadir point and the ideal point have only finite components each. The objectives are scaled by an affine transformation such that the ideal point has coordinates \(z^I = (0, \ldots, 0)^T\) and the nadir point has coordinates \(z^N = (1, \ldots, 1)^T\).

2.3. Measures of approximation quality

In multicriteria optimization, a large number of measures of the approximation quality exist [26]. For obtaining convergence results the choice of a measure with good theoretical properties is key.

2.3.1. Difference volume

The difference volume aims to quantify the average error that the approximation has to the true Pareto frontier. We use this as the basis for both the construction and the analysis of our algorithm.

Definition 2.2 (Difference volume). Let \(A \subseteq Z\) be an approximation of the set of feasible objective vectors \(Z\). Note that by our scaling assumption, it holds \(Z \subseteq [0, 1]^K\). The difference volume \(\delta_{\text{vol}}(Z, A)\) of \(Z\) and \(A\) with respect to the standard ordering cone \(\mathbb{R}^K_{\geq 0}\) is defined as the enclosed volumes between the sets completed by their ordering cones:

\[
\delta_{\text{vol}}(Z, A) := \text{vol}\left(\left((Z + \mathbb{R}^K_{\geq 0}) \setminus (A + \mathbb{R}^K_{\geq 0})\right) \cap [0, 1]^K\right) - \text{vol}\left((A + \mathbb{R}^K_{\geq 0}) \cap [0, 1]^K\right)
\]

The difference volume is closely related to the hypervolume-measure [27] as it can be represented as the difference in hypervolume of the approximation and the true Pareto frontier. In [28] the related hyperarea difference is defined which is equivalent to the difference volume if a matching scaling is used.
**Figure 4.** Illustration of ε-indicator value and difference volume for an approximation A (thick black line) of the true Pareto frontier Z (the boundary of the shaded area). The darker shaded area represents the difference volume $\delta_{\text{vol}}(Z,A)$. The ε-indicator value $\varepsilon_d(Z,A)$ is illustrated with the dotted line.

**Integral representation of the difference volume.** In our main case of $K = 2$ criteria the difference volume can be also represented as an integral. The formulation is based on the smallest $y$-coordinate $B_y(x)$ of points in a set $B \subseteq \mathbb{R}^2$ with $x$-coordinate being at most a given value $x$, formally defined as

$$B_y(x) := \min\{y' \mid (x, y') \in B + \mathbb{R}^2_\geq 0\}.$$  

(1)

The difference volume of an approximation $A$ can be represented by

$$\delta_{\text{vol}}(Z,A) = \int_0^1 (A_y(x) - Z_y(x)) \, dx.$$  

(2)

This formulation leads to an interpretation of the difference volume as the average error of the approximation: If an upper bound for the first objective is chosen uniformly at random in $[0,1]$, taking the point with minimal value in the second objective fulfilling this bound from the approximation $A$ instead of the exact set $Z$ will lead to an average deviation in the second objective of $\delta_{\text{vol}}(Z,A)$.

2.3.2. The ε-indicator

For comparison purposes we also use the ε-indicator [29]. This measure has been widely used in different forms throughout the literature, sometimes under the notion ε-approximate Pareto frontier [30,31].

**Definition 2.3 (ε-indicator).** The ε-indicator value $\varepsilon_d(Z,A)$ of an approximation $A \subseteq Z$ is given as

$$\varepsilon_d(Z,A) := \sup_{z \in Z + \mathbb{R}^K_\geq 0} \inf_{x \in A + \mathbb{R}^K_\geq 0} ||z - x||_\infty = \sup_{z \in Z} \inf_{x \in A + \mathbb{R}^K_\geq 0} ||z - x||_\infty$$

where we use the Chebyshev norm

$$||v||_\infty := \max\{|v_1|, |v_2|, \ldots, |v_K|\}.$$
We use this measure in our numerical analysis to show that our algorithm also obtains a reasonable convergence with respect to other approximation quality measures. The measure is computed using the methods in [32].

2.4. Patch complexity

To describe the inherent complexity of a multicriteria optimization problem, we use the concept of the patch complexity. This will give us a lower bound for the number of patches required to get an approximation with some given quality. We base the analysis of our algorithm on a comparison to this instance-specific lower bound.

Definition 2.4 (Segment patch complexity). The segment patch complexity of a multicriteria optimization problem with a set of feasible objective vectors \( Z \) with respect to the approximation quality measure \( q \) at level \( \varepsilon > 0 \) is given by the minimal number of patches that are needed to get an approximation \( A \subseteq Z \) such that \( q(A,Z) \leq \varepsilon \).

Formally, the segment patch complexity \( N^S_{\delta_{vol}} \) with respect to the difference volume is given as the minimal number \( n \in \mathbb{N} \) of segment patches \( s_1, \ldots, s_n \) such that, for the union \( A = \bigcup_{i=1}^n s_i \), it holds \( \delta_{vol}(A,Z) \leq \varepsilon \).

Our algorithm finds a set of patches with approximately minimal cardinality such that the difference volume is bounded by a desired value. The authors of [15] follow the similar goal to find a set of points with minimal cardinality such that an approximation error to the Pareto frontier is bounded. They give algorithms that compute point sets that are at most a constant factor away from the optimum, an alternative algorithm for this is given in [33]. Our use of patches instead of just points however significantly reduces the required number of parts of the Pareto frontier to obtain a given approximation quality.

3. The Adaptive Patch Approximation Algorithm

We now describe on a high-level our Adaptive Patch Approximation Algorithm that combines patches to an approximate Pareto frontier of a bicriteria mixed-integer program.

3.1. The iterative improvement approach

Our method is based on the idea to adaptively improve the current approximation with segment patches that decrease the difference volume as much as possible. In each iteration, we compute a segment patch that fulfills the improvement guarantee below, i.e. the addition of the patch should improve the difference volume of the approximation as much as possible. The resulting segment patch will then be added to the approximation. The details of how such a segment patch can be found efficiently will be discussed in section 4.

We denote the \( y \)-coordinate of a segment patch \( s \) at a given \( x \)-coordinate \( x \in [X_1(s),X_2(s)] \) by \( s_y(x) \). We also use the smallest \( y \)-coordinate \( A_y(x) \) of a point in \( A \) with given \( x \)-coordinate defined in [1]. The analysis of the improvement in difference volume obtained by a segment patch is based on the following measure that is better suited for optimization.

Definition 3.1 (Integrated improvement). For an approximation \( A \) and a segment patch \( s \), the Integrated improvement \( I_A(s) \) by \( s \) w.r.t. the approximation \( A \) is given by the volume between
the approximation and the segment

\[
I_A(s) = \int_{X_1(s)}^{X_2(s)} (A_y(x) - s_y(x)) \, dx.
\]

Optimizing with respect to the Integrated improvement will allow us to make progress on the difference volume due to the following relation.

**Proposition 3.1** (Impact of Integrated improvement on difference volume). *Let \( A \) be some approximation of the set \( Z \) to which the segment patch \( s \) is added. Then the new difference volume is bounded by*

\[
\delta_{\text{vol}}(Z, A \cup s) \leq \delta_{\text{vol}}(Z, A) - I_A(s).
\]

**Proof.** Using the integral representation (2) of the difference volume and the bound \((A \cup s)_y(x) \leq s_y(x)\) for \( x \in [X_1(s), X_2(s)]\), the difference volume of the extended approximation \( A \cup s \) can be bounded by

\[
\delta_{\text{vol}}(Z, A \cup s)
\leq \int_{X_1(s)}^{X_1(s)} (A_y(x) - Z_y(x)) \, dx + \int_{X_1(s)}^{X_2(s)} (s_y(x) - Z_y(x)) \, dx + \int_{X_2(s)}^{X_2(s)} (A_y(x) - Z_y(x)) \, dx.
\]

By cancelling shared terms with the integral representation (2) of \( \delta_{\text{vol}}(Z, A) \) we get the bound

\[
\delta_{\text{vol}}(Z, A) - \delta_{\text{vol}}(Z, A \cup s)
\leq \int_{X_1(s)}^{X_2(s)} (A_y(x) - Z_y(x)) \, dx - \int_{X_1(s)}^{X_2(s)} (s_y(x) - Z_y(x)) \, dx
\]

\[
= \int_{X_1(s)}^{X_2(s)} (A_y(x) - s_y(x)) = I_A(s)
\]

from which we obtain the claim. \( \square \)

In each iteration, we aim to find a segment patch that provides a large Integrated improvement with respect to the current approximation and thus a large progress in the difference volume. Obtaining the segment patch with maximal Integrated improvement is cumbersome. However, guaranteeing that the Integrated improvement of the segment patch is within some factor of the maximally possible value is sufficient for obtaining a good convergence. We show this by analysing the effect of this guarantee in detail.

**Definition 3.2** (Integrated improvement guarantee). An algorithm for computing a segment patch fulfils the Integrated improvement guarantee at level \( \kappa > 0 \) if on each approximation \( A \) it returns a segment patch \( s \) such that

\[
I_A(s) \geq \kappa \max_{s' \text{segment patch}} I_A(s'),
\]

i.e. the Integrated improvement provided by the new segment patch to the old approximation should be at least a factor \( \kappa \) times the maximally possible Integrated improvement for a feasible segment patch.
We now have all prerequisites to formulate our main algorithm, the Adaptive Patch Approximation [Algorithm 1]. On a high level, the Adaptive Patch Approximation Algorithm iteratively adds segment patches to the approximation, until a stopping criterion is reached.

**Main Algorithm 1: Adaptive Patch Approximation algorithm**

\[
A \leftarrow \emptyset \\
\textbf{while} \text{ stopping criterion not reached} \textbf{do} \\
\quad \text{Find a segment patch } s \text{ fulfilling the Integrated improvement guarantee (for some constant level } \kappa) \text{ to the current approximation } A \\
\quad A \leftarrow A \cup s
\]

In the following sections we establish the efficiency of this algorithm by analysing its convergence and describing an effective realization. We show that the convergence rate almost matches the theoretical optimum given by the patch complexity. In section 4 we describe how the Integrated improvement guarantee can be fulfilled using only a constant number of optimization steps per iteration.

### 3.2. Ingredients for the analysis: Lower Envelopes and Pareto frontiers

For our analysis of the convergence rate we use monotone lower envelopes. For a set of segment patches, the monotone lower envelope corresponds to the relevant, nondominated parts. We can hence use this as a simple geometric model for the effects of the insertion of a segment patch into the approximation.

Recalling standard computational geometry, the lower envelope of line segments \(s_1, \ldots, s_n\) with respect to the \(x\)-axis is given as the following curve

\[
\ell(x) = \min\{y \in \mathbb{R} \mid (x, y) \text{ is contained in } s_i \text{ for some } i \in \{1, \ldots, n\}\}.
\]

To properly model the set of nondominated points, we use the following modified version of the lower-envelope that ensures monotonicity to match the behavior of Pareto frontiers.

**Definition 3.3 (Monotone lower envelope).** The monotone lower envelope of a set of line segments \(s_1, \ldots, s_n\) is given as

\[
m(x) = \min\{y \in \mathbb{R} \mid \exists x' \leq x \text{ s.t. } (x', y) \text{ is contained in } s_i \text{ for some } i \in \{1, \ldots, n\}\}.
\]

Figure 5 shows an example of the monotone lower envelope. Our approximation quality measures are invariant under taking the monotone lower envelope since it corresponds to the addition of the ordering cone \(\mathbb{R}^K_{\geq 0}\).

**Proposition 3.2 (Invariance of measures under monotone lower envelope).** Let \(A = \bigcup_{i=1}^n s_i\) be an approximation consisting of a set of segments \(s_1, \ldots, s_n\). Let \(A'\) denote the monotone lower envelope of \(s_1, \ldots, s_n\). Then the following equalities hold:

- \(N(A) = N(A')\)
- \(\delta_{vol}(A) = \delta_{vol}(A')\)
- \(\epsilon_d(A) = \epsilon_d(A')\)

For brevity we skip the proof. The result shows that we can work with the monotone lower envelope without affecting our guarantees on the approximation quality.
Figure 5. Illustration of the monotone lower envelope \( m \) (gray line) for a set of segments (dashed). The dotted line shows the difference in value for the monotone lower envelope \( m \) and the standard lower envelope \( \ell \) at the position \( x' \).

Complexity of monotone lower envelopes. For our convergence guarantees we need a bound on the number of segments contained in the monotone lower envelope.

We denote the standard lower envelope complexity, i.e. the maximal number of segments occurring in the lower envelope of \( n \) segments in the plane, by \( \lambda(n) \). The following classic bound from computational geometry limits this standard lower envelope complexity to an almost-linear growth.

**Lemma 3.1** ([34][35]). For \( n \in \mathbb{N} \) the standard lower envelope complexity is bounded by

\[
\lambda(n) \leq 2n \ln n
\]

and also by

\[
\lambda(n) \leq 52n(\alpha(n) + 1)
\]

where \( \alpha \) is the inverse Ackermann function, a very slowly growing function.

As a corollary we obtain the following bound on the maximal number of segments in a monotone lower envelope which we denote by \( \bar{\lambda}(n) \).

**Corollary 3.1** (Complexity of monotone lower envelopes). Let a set \( S \) of \( n \) line segments be given. Then the monotone lower envelope of these segments consists of at most

\[
\bar{\lambda}(n) \leq \lambda(2n) \leq \min\{4n \ln(2n), 104n(\alpha(2n) + 1)\}
\]

many segments.

**Proof.** We can obtain the monotone lower envelope in the following way: We add for each segment \( s \in S \) an additional horizontal segment \( s' \) extending from the right endpoint of \( s \) until the largest \( x \)-coordinate in \( S \). Let \( S' \) be this extended set of segments. Due to the extensions with the horizontal segments, the lower envelope of \( S' \) is equal to the monotone lower envelope of \( S \). The number of segments of the monotone lower envelope of \( S \) is thus bounded by the number of segments in the standard lower envelope of \( S' \). Since \( |S'| \leq 2|S| \) we thus obtain from applying [Lemma 3.1] on \( S' \) the upper bound

\[
\bar{\lambda}(n) \leq \lambda(|S'|) \leq \lambda(2|S|) \leq \min\{4n \ln(2n), 104n(\alpha(2n) + 1)\}
\]
for the number of segments in the monotone lower envelope.

Note that both functions $\lambda(n)$ and $\bar{\lambda}(n)$ are monotonically increasing since by adding a small segment at an appropriate place the number of segments in the lower envelope can always be increased.

### 3.3. Convergence bound

Using monotone lower envelopes, we now show that the Adaptive Patch Approximation Algorithm 1 achieves a convergence with respect to the difference volume which is only a small factor away from the theoretical optimum given by the patch complexity.

**Lemma 3.2 (Consequence of Improvement guarantee).** Let $Z$ be the set of all feasible objective vectors and $A \subseteq Z$ the current approximation. Suppose that we generate the next approximation $A'$ by adding a segment patch fulfilling the Integrated improvement guarantee with level $\kappa > 0$. Then, the new difference volume $\delta_{\text{vol}}(Z,A')$ is bounded by

$$
\delta_{\text{vol}}(Z,A') \leq \delta_{\text{vol}}(Z,A) - \frac{\kappa}{2\bar{\lambda}(N_{S}(\delta_{\text{vol}}(Z,A)/2))}.
$$

**Proof.** We show the statement by establishing the bound

$$
\delta_{\text{vol}}(Z,A') \leq \delta_{\text{vol}}(Z,A) - \frac{\delta_{\text{vol}}(Z,A) - \varepsilon}{\bar{\lambda}(N_{S}(\varepsilon))} \quad \text{for all } \varepsilon > 0. \quad (3)
$$

The claim of the lemma then follows from plugging in $\varepsilon = \delta_{\text{vol}}(Z,A)/2$. For $\varepsilon \geq \delta_{\text{vol}}(Z,A)$ the bound (3) follows immediately, since $\delta_{\text{vol}}(Z,A') \leq \delta_{\text{vol}}(Z,A)$ holds due to the monotonicity of the difference volume.

Let some $\varepsilon < \delta_{\text{vol}}(Z,A)$ be given. By the definition of the segment patch complexity $N_{S}(\varepsilon)$ there exists a set $S'$ of $|S'| = N_{S}(\varepsilon)$ many segment patches such that $\delta_{\text{vol}}(Z,S') \leq \varepsilon$. Let $S'$ be the monotone lower envelope of $S'$. It holds $\delta_{\text{vol}}(Z,S') = \delta_{\text{vol}}(Z,S')$ by Proposition 3.2. Because $S'$ is a monotone lower envelope, the segments of $S'$ cover the whole range $[0,1]$ on the $x$-axis without overlapping. We can thus represent the difference volume $\delta_{\text{vol}}(Z,S') \leq \varepsilon$ as

$$
\delta_{\text{vol}}(Z,S') = \delta_{\text{vol}}(Z,S') = \sum_{\text{segment } s' \text{ in } S'} \int_{X_{1}(s')}^{X_{2}(s')} (s'_{y}(x) - Z_{y}(x)) \, dx
$$

using the representation (2) by a comparison to the true Pareto frontier given by $Z$. Using the same partition of the $x$-axis given by these segments, we can represent the current difference volume as

$$
\delta_{\text{vol}}(Z,A) = \sum_{\text{segment } s' \text{ in } S'} \int_{X_{1}(s')}^{X_{2}(s')} (A_{y}(x) - Z_{y}(x)) \, dx.
$$
By taking the difference of the integrals, we arrive at the inequality

$$\delta_{\text{vol}}(Z,A) - \varepsilon \leq \delta_{\text{vol}}(Z,A) - \delta_{\text{vol}}(Z,S') = \sum_{\text{segment } s' \text{ in } S'} \int_{X_1(s')} \left( A_y(x) - s'_y(x) \right) dx.$$

Since $\varepsilon < \delta_{\text{vol}}(Z,A)$ holds and because for at least one of the summands a value not smaller than the average value must occur, there exists a segment $s^* \in S'$ that fulfills the inequality

$$I_A(s^*) = \int_{X_2(s^*)} \left( A_y(x) - s^*_y(x) \right) dx \geq \frac{1}{|S'|} \left( \delta_{\text{vol}}(Z,A) - \varepsilon \right).$$

Let $s$ be the new segment patch added to the approximation. Since $s$ fulfills the Integrated improvement guarantee, the improvement obtained by $s$ is at least within a factor $\kappa$ of that obtained by $s^*$. Thus we have

$$I_A(s) \geq \kappa I_A(s^*) \geq \frac{\kappa}{|S'|} \left( \delta_{\text{vol}}(Z,A) - \varepsilon \right) \geq \frac{\kappa}{\lambda(N_{\delta_{\text{vol}}}^S(\varepsilon))} \left( \delta_{\text{vol}}(Z,A) - \varepsilon \right),$$

where we used the fact that the set $S'$ contains at most $|S'| \leq \lambda(N_{\delta_{\text{vol}}}^S(\varepsilon))$ many segments.

According to Proposition 3.1, by adding the segment patch $s$ the difference volume $\delta_{\text{vol}}$ decreases by at least $I_A(s)$. We thus obtain the statement by choosing $\varepsilon$ arbitrarily and using the bound on $I_A(s)$.

We are now able to show the main convergence result of our algorithm: The number of iterations required to reach a difference volume of $\varepsilon$ is close to the theoretical optimum $N_{\delta_{\text{vol}}}^S(\varepsilon)$.

**Theorem 3.1 (Convergence of the Adaptive Patch Approximation Algorithm).** Let $\varepsilon \in (0, 1)$ be an arbitrary bound on the difference volume. Then, the approximation returned by the Adaptive Patch Approximation Algorithm 1 reaches a difference volume of at most $\varepsilon$ after at most

$$\frac{2\lambda(N_{\delta_{\text{vol}}}^S(\varepsilon)/2)}{\kappa} \ln(1/\varepsilon)$$

many iterations.

**Proof.** Let $A_0, \ldots, A_k$ be the sequence of approximations generated by the iterations of the algorithm, where $A_0 = \emptyset$ is the start approximation and $A_k$ is the first approximation where the requirement $\delta_{\text{vol}}(Z,A_k) \leq \varepsilon$ is met. Thus, it holds $\delta_{\text{vol}}(Z,A_\ell) > \varepsilon$ for all $\ell \in \{0, \ldots, k-1\}$. Since the patch complexity is a monotonically decreasing function, this implies

$$N_{\delta_{\text{vol}}}^S(\delta_{\text{vol}}(Z,A_\ell)/2) \leq N_{\delta_{\text{vol}}}^S(\varepsilon/2) \quad \text{for all } \ell \in \{0, \ldots, k-1\}.$$

Lemma 3.2 hence yields for each iteration $\ell \in \{0, \ldots, k-1\}$ the following bound for the
As $\delta_{\text{vol}}(Z,A_0) = \delta_{\text{vol}}(Z,0) \leq 1$ by the definition of the difference volume, we get the total bound

$$\delta_{\text{vol}}(Z,A_k) \leq \left(1 - \frac{\kappa}{2\bar{\lambda}\left(N^S_{\delta_{\text{vol}}}(\varepsilon/2)\right)}\right)^k.$$  

Thus, to achieve $\delta_{\text{vol}}(Z,A_k) \leq \varepsilon$, at most

$$k \leq \ln(\varepsilon) \cdot \left(\ln\left(1 - \frac{\kappa}{2\bar{\lambda}\left(N^S_{\delta_{\text{vol}}}(\varepsilon/2)\right)}\right)\right)^{-1}$$  

many iterations are needed. By using the inequality $\ln(1 - x) \leq -x$ for $x < 1$, we obtain from above

$$k \leq \ln(\varepsilon) \cdot \left(-\frac{\kappa}{2\bar{\lambda}\left(N^S_{\delta_{\text{vol}}}(\varepsilon/2)\right)}\right)^{-1} = 2\ln(1/\varepsilon) \cdot \frac{\bar{\lambda}\left(N^S_{\delta_{\text{vol}}}(\varepsilon/2)\right)}{\kappa}$$

as claimed, where we used $\ln(\varepsilon) < \ln(1) = 0$.  

4. Efficiently computing segment patches

In the preceding section we have shown that the Adaptive Patch Approximation Algorithm 1 obtains a near-optimal convergence with respect to the number of iterations. In this section we describe how the individual iterations can be implemented. The key task here is to develop an algorithm that efficiently finds a good segment patch. In particular, we have to ensure that the improvement guarantee of Definition 3.2 is fulfilled without requiring too many optimizations of the underlying mixed-integer program.

Since we build our approximation $A$ out of segment patches, the approximation can be described by the monotone lower envelope of the set of segment patches. We partition the objective space along the axis of the first objective $f_1$ according to these segments in the monotone lower envelope. In the following, we identify the $x$-axis with the value of the first objective function $f_1$. In this way, vertical regions are created. Figure 6 illustrates this partition of the objective space.

Our strategy for obtaining good segment patches is based on searching for segment patches that are fully contained in one of the regions induced by the monotone lower envelope of the approximation. When searching a segment patch $s'$ with respect to segment $s$, we require
that the bounds on the \( x \)-coordinates of the endpoints \( X_1(s) \leq X_1(s') \leq X_2(s') \leq X_2(s) \) are fulfilled. The restriction to such a region ensures that the approximation is linear with respect to \( x \) which enables a simple formulation of the desired optimization goal. We can reuse most found patches, only for segments that are new in the monotone lower envelope a new patch needs to be computed. In each iteration, we then add the patch that achieves the maximal improvement in difference volume among all candidate patches.

The realization of our Adaptive Patch Approximation algorithm can be summarized as follows in Main Algorithm 2. As a step to ensure approximate optimality in each iteration, we use Algorithm 3 which is described later.

**Main Algorithm 2: Realization of the Adaptive Patch Approximation algorithm**

1. Initialize the approximation: \( A \leftarrow \emptyset \)
2. Initialize the monotone lower envelope: \( M \leftarrow \{ \text{segment from } (0,1) \text{ to } (1,1) \} \)
3. Initialize the list of candidate patches: \( P \leftarrow \emptyset \)
4. While \( \delta_{vol}(Z,A) > \varepsilon \) do:
   - foreach new segment \( m \in M \) of the monotone lower envelope do:
     - Find a segment patch in the region induced by \( m \) with maximal Integrated improvement w.r.t. \( m \) (approximately to a factor \( 1 - \beta \))
     - Add the segment patch to \( P \)
   - Take the segment patch \( p^* \in P \) with maximal difference volume improvement with respect to \( A \)
   - Re-optimize with respect to the coordinates of this patch \( p^* \) to ensure local optimality with Algorithm 3 and update \( p^* \) if needed
   - Update the approximation: \( A \leftarrow A \cup p^* \)
   - Update the monotone lower envelope \( M \) of \( A \)

**Number of segment patch computations per iteration.** In each iteration of the algorithm, the monotone lower envelope \( M \) of \( A \) has to be updated after insertion of a new segment patch. However, we show that the number of changed segments in \( M \) is at most 4 per iteration. This follows from the fact that we add only segment patches to the approximation that have been contained in a region given by a previous segment. Thus, when adding a new segment patch \( p \) inside the region given by the old segment \( s \) from the monotone lower envelope, only two cases can occur which are also illustrated in Figure 7.
(a) The new segment patch \( p \) is above the right endpoint of the segment \( m \), i.e. has a larger \( y \)-coordinate everywhere. Then, at most 4 new segments can appear in the monotone lower envelope: A part of \( p \), two parts of \( s \) (one the left and one to the right of \( p \)) and a horizontal segment.

(b) The new segment \( p \) is (at least partially) below the right endpoint of \( s \). Then, there is only one part of \( s \) and one part of \( p \) that might change. Additionally, only the horizontal segment extending from the right endpoint of \( p \) and the segment intersecting this horizontal segment can be created or updated. Thus, at most 4 segments are updated. There might however be segments from the monotone lower envelope that will be completely removed by the insertion of \( p \).

Our discussion shows that at most 4 segments are updated in the monotone lower envelope per iteration. This results in the following bound on the effort required by Algorithm 2.

**Proposition 4.1** (Complexity of the Adaptive Patch Approximation Algorithm). The Adaptive Patch Approximation [Algorithm 2] needs to compute at most 4 segment patches per iteration.

In the remainder of this section, we show the following statement that establishes the effectiveness and efficiency of Algorithm 2.

**Theorem 4.1.** The Adaptive Patch Approximation [Algorithm 2] has the following properties for every fixed value of the parameter \( \beta > 0 \):

- A segment patch can be found by optimizing a MIP of similar form as the original problem. This MIP has double the number of constraints and continuous variables as the original MIP, with an additional constant number of constraints and variables.
- The patch improvement guarantee is always fulfilled with \( \kappa = \frac{1}{8}(1 - \beta) \).

To prove this theorem, we perform a detailed analysis of the various steps of the algorithm.

4.1. Finding good segment patches by maximizing the Integrated improvement

In the Adaptive Patch Approximation algorithm, we need to find segment patches with maximal Integrated improvement with respect to the current segments. We now give an efficient MIP formulation for this task.
Expression for the Integrated improvement. The basis of our formulation is a simple expression for the Integrated improvement w.r.t. a single segment. Assume that the new segment patch \( s' \) is found inside the region induced by the segment \( s \) of the current monotone lower envelope, i.e. it holds \( X_1(s) \leq X_1(s') \leq X_2(s') \leq X_2(s) \). The Integrated improvement \( I_s(s') \) can then be simplified to

\[
I_s(s') = \int_{X_1(s')}^{X_2(s')} (s_x(x) - s'_x(x)) \, dx
\]

\[
= \left( X_2(s') - X_1(s') \right) \cdot \left( Y_1(s) + \frac{Y_2(s) - Y_1(s)}{X_2(s) - X_1(s)} \cdot \frac{X_1(s')/2 + X_2(s')/2 - X_1(s)}{2} \right)
\]

by using the linearity of the segments \( s \) and \( s' \).

Formulation of the MIP. Our implementation of the Adaptive Patch Approximation [Algorithm 1] is based on computing, with respect to each segment of the monotone lower envelope, a segment patch with approximately maximal Integrated improvement. We now show how such an optimization can be formulated in practice. For this, an approximately optimal solution is sufficient. We will show that we can obtain an approximation with a constant relative error that can be chosen to be as close to 0 as wanted.

To maximize the Integrated improvement corresponding to the reference segment \( s \), we have to solve the following optimization problem.

\[
\max I_s(s^*)
\]

s. t. \( s^* \) is a segment patch in objective space

\[
X_2(s^*) \geq X_1(s^*)
\]

\[
X_1(s^*) \geq X_1(s)
\]

\[
X_2(s^*) \leq X_2(s)
\]

We use here [Definition 2.1] of segment patches in objective space. Note that according to this definition an extension in direction of the ordering cone is allowed and thus horizontal segment patches can also be considered. The requirement that \( s^* \) is a segment patch can be explicitly modelled using linear inequalities and variables and constraints from the original mixed-integer program. By additionally using the expression (4) for the Integrated improvement \( I_s(s^*) \) we arrive at the following formulation (5) to compute an optimal segment patch with endpoints \((x_1,y_1)\) and \((x_2,y_2)\).
The optimization objective in (5) is nonlinear. To maintain the linear structure of the original MIP, we now linearize this objective. Using the linear substitutions

\[ x = (x_2 - x_1) \]

and

\[ y = Y_1(s) + (Y_2(s) - Y_1(s)) \frac{X_2(s) - X_1(s)}{X_2(s) - X_1(s)} - \frac{y_1 + y_2}{2} \]

we get the following simple bilinear term for \( I(s') \) which is equivalent to (5)

\[ I(s') = g(x, y) := x \cdot y. \]

For this function \( g \) we perform now an approximate linearization. Note that by definition \( x = x_2 - x_1 \geq 0 \) is non-negative. Since the maximal value of \( I(s') \) is at least 0 we can assume \( y \geq 0 \) and include it as a constraint in the problem. We thus only need to approximate the function \( g \) in the non-negative orthant. Due to the scaling of the objective vectors to \([0, 1]^2\), we also know that the \( x \)- and \( y \)-value can be at most 1. Thus, approximating \( g \) in the square \([0, 1]^2\) is sufficient. We use an approach based on the identity

\[ g(x, y) = x \cdot y = \exp(\ln(x) + \ln(y)). \]

Since \( \exp \) is a monotone function, maximizing \( g \) is equivalent to maximizing \( \ln(x) + \ln(y) \). For the logarithm we can give an upper approximation with linear inequalities based on tangents, since the logarithm is a concave function.

**Approximation of the logarithm with linear inequalities.** To provide a linearization of the Integrated improvement obtained by a segment patch, we give an approximation of \( \ln(x) \)
for values $x \in [\delta_0, 1]$ larger than some constant threshold $\delta_0 > 0$. It is sufficient to restrict the approximation to this range, because smaller values of $x$ correspond to an integrated improvement which is guaranteed to be smaller than available alternatives. Our approach is based on tangents with an exponentially scaled discretization.

A similar problem of approximating a convex function via a convex piecewise linear approximation was considered in [36]. For the function $\ln(1 + e^x)$ it was shown that there is an optimal approximation consisting of tangents at appropriate breakpoints which can be found with an algorithm. We describe here a similar procedure, which given some wanted additive accuracy $\beta' > 0$ finds a concave piecewise linear approximation of $\ln(x)$ with a minimal number of pieces.

Formally, we search a set of $k$ lines described by the slope $a_i \in \mathbb{R}$ and $y$-intercept $b_i \in \mathbb{R}$ for $i = 1, \ldots, k$, such that

$$\min_{i=1,\ldots,k} a_i x + b_i - \beta' \leq \ln(x) \leq \min_{i=1,\ldots,k} a_i x + b_i \quad \text{for all } x \in [\delta_0, 1].$$

This form allows us to later transform a maximization of $\ln(x) + \ln(y)$ into the approximately equivalent problem

$$\max z_x + z_y$$

s. t. $z_x \leq a_i x + b_i$ for each $i \in \{1, \ldots, k\}$

$$z_y \leq a_i y + b_i$$

for each $i \in \{1, \ldots, k\}$,

with a linear structure. The approximation of $\ln(x)$ and correspondingly $\ln(y)$ within an additive error of $\beta'$ yields a total relative error for the approximation of $x \cdot y$ of

$$\beta = 1 - \exp(-2\beta') \in (0, 1)$$

as a simple calculation shows. With our construction we will establish the following statement:

**Lemma 4.1 (Approximation of the logarithm with linear inequalities).** Let $\beta' > 0$ and $\delta_0 > 0$ be given. Then, there exists a set of lines of size

$$k \leq \frac{\ln(1/\delta_0)}{\ln(1 + \sqrt{2\beta'})}$$

which are given by the slopes $a_i \in \mathbb{R}$ and $y$-intercepts $b_i \in \mathbb{R}$, $i = 1, \ldots, k$, such that

$$\min_{i=1,\ldots,k} a_i x + b_i - \beta' \leq \ln(x) \leq \min_{i=1,\ldots,k} a_i x + b_i \quad \text{for all } x \in [\delta_0, 1].$$

This lemma implies immediately, by replacing the value $b_i$ by $b_i - \beta'$, that we can also get the similar approximation from below

$$\min_{i=1,\ldots,k} a_i x + b_i \leq \ln(x) \leq \min_{i=1,\ldots,k} a_i x + b_i + \beta' \quad \text{for all } x \in [\delta_0, 1].$$

The construction is based on the idea that each line of the approximation should be a tangent to $\ln(x)$. Choosing tangents is optimal, since any approximation that is not directly touching
the graph of the logarithm can be improved by shifting and secants are not allowed because we want to overestimate the function \( \ln(x) \).

In Appendix A, the construction is described in detail and a proof of Lemma 4.1 is provided. Figure 8 shows the corresponding approximation for the case \( \beta' = 0.2 \) and \( \delta_0 = 0.022 \).

![Figure 8. Approximation of the logarithm (blue line) via linear inequalities (dashed lines). The shown plot corresponds to the choice \( \beta' = 0.2 \). With the tangents through the three points \( x_0, x_1, x_2 \) we obtain an upper approximation of \( \ln(x) \) with an additive error of at most \( \beta' = 0.2 \) for the interval \( [\delta_0, 1] = [z_3, 1] \approx [0.022, 1] \) which is attained at the points \( z_1, z_2 \) and \( z_3 \).](image)

4.2. Establishing optimality of the algorithm

We show now that the Adaptive Patch Approximation Algorithm 2 fulfills the Integrated improvement guarantee with a factor of \( \kappa = \frac{1}{8}(1 - \beta) \). This result then implies the almost-optimal convergence rate of the algorithm as shown in Theorem 3.1. To establish the Integrated improvement guarantee, we have to show that the segment patch chosen by our algorithm obtains an Integrated improvement that is at least \( \frac{1}{8}(1 - \beta) \) as large as the maximally possible Integrated improvement for an arbitrary segment patch. Our proof for this is based on the decomposition of the Integrated improvement obtained by the optimal segment patch into the regions given by the monotone lower envelope. We bound the number of regions for which a positive Integrated improvement occurs.

Notation for segment relations. New segment patches are found by optimizing with respect to given segments of the monotone lower envelope. The tree-like relations between these segments are the basis of our analysis. We denote the parent-segment of a segment \( s' \) in the monotone lower envelope, i.e. the segment in which region a segment patch was found that lead to the creation of \( s' \), by \( \text{par}(s') \). The set of parent-segments of an approximation \( A \) consists of the parent-segments of each of the segments in the corresponding monotone lower envelope. Reversely, the children of \( s \) are the segments that resulted from an optimization within the region of \( s \). The set of children of \( s \) is denoted by \( \text{chldn}(s) \). As shown in the discussion of Proposition 4.1, the number of children of a segment is at most 4.

Local optimality guarantee. Due to the linearization, we obtain only segment patches that maximize within a factor of \( (1 - \beta) \) the Integrated improvement possible inside a region. Thus, we cannot directly guarantee exact optimality, which is required for our decomposition.
approach. We can use instead the following local guarantee which is simpler to ensure.

**Definition 4.1 (Local optimality guarantee).** Let \( s^* \) be a new segment patch that was returned by the algorithm and \( \text{par}(s^*) \) its corresponding parent segment. The segment \( s^* \) fulfils the local optimality guarantee if for every other segment patch \( s' \) with range

\[ [X_1(s'), X_2(s')] = [X_1(\text{par}(s^*)), X_2(\text{par}(s^*))] \]

the Integrated improvement is not larger, i.e.

\[ I_{\text{par}(s^*)}(s') \leq I_{\text{par}(s^*)}(s^*). \]

As the guarantee only concerns fixed \( x \)-coordinates the Integrated improvement \( I_{\text{par}(s^*)}(s') \) becomes linear in the \( y \)-coordinates. Thus the guarantee can be easily provided as a post-optimization step by Algorithm 3.

**Algorithm 3: Ensuring local optimality**

**Input:** a reference segment \( s \) along with a candidate \( s^* \) for the corresponding optimal child segment  
**Output:** a segment patch fulfilling the local optimality guarantee  
Compute a segment patch \( s' \) with fixed coordinates \( X_1(s') = X_1(s), X_2(s') = X_2(s) \)  
with minimal \( Y_1(s') + Y_2(s') \)  
if \( I_s(s') > I_s(s^*) \) then  
    return \( s' \)  
else  
    return \( s^* \)

As a consequence of the local optimality guarantee we can rule out that a positive Integrated improvement is possible throughout the whole region of the parent segment. We use here the the extended form \( \tilde{I}_s(s') \) of the Integrated improvement of \( s' \) with respect to a segment \( s \) for the case that \( s \) does not cover the whole \( x \)-range of \( s' \). In this situation, the integral is restricted to the overlapping \( x \)-range between the segments as

\[ \tilde{I}_s(s') := \int_{\max\{X_1(s), X_1(s')\}}^{\min\{X_2(s), X_2(s')\}} (s_y(x) - s'_y(x)) \, dx. \]

In the case that the segment \( s' \) is covered by the reference segment \( s \), i.e. if \( X_1(s) \leq X_1(s') \leq X_2(s') \leq X_2(s) \) holds, this extended Integrated improvement is equivalent to the original \( \tilde{I}_s(s') = I_s(s') \).

**Lemma 4.2 (No further improvement over the whole interval).** Assume that the local optimality guarantee is fulfilled for \( s^* \). Then, any new segment patch \( s' \) extending over the whole \( x \)-interval of \( \text{par}(s^*) \)

\[ [X_1(s'), X_2(s')] = [X_1(\text{par}(s^*)), X_2(\text{par}(s^*))] \]

does not obtain a positive Integrated improvement over this interval, i.e.

\[ \sum_{s' \in \text{child}(\text{par}(s^*))} \tilde{I}_{s'}(s') \leq 0. \]
Proof. Let \( s' \) be any segment patch fulfilling the assumption. Since this segment patch extends over the whole region induced by \( \text{par}(s') \), the Integrated improvement of \( s' \) can be decomposed as

\[
\sum_{s'' \in \text{chldn}(\text{par}(s'))} I_{s''}(s') = I_{\text{par}(s')}(s') - \sum_{s'' \in \text{chldn}(\text{par}(s'))} I_{\text{par}(s')}(s'')
\]

where we used the local optimality guarantee for the first inequality. \( \square \)

Obtaining the Integrated improvement guarantee. The following decomposition of the Integrated improvement \( I_{A}(s') \) of an arbitrary segment patch shows that the best segment patch restricted to a region of the approximation obtains at least \( \frac{1}{8} \) of that Integrated improvement. Thus searching for segment patches inside the individual regions will yield sufficiently good results.

Lemma 4.3 (Separate optimization in each segment is sufficient). Let \( s' \) be an arbitrary segment patch. Assume that the local optimality guarantee holds for all parent segments of the current approximation \( A \) and corresponding monotone lower envelope \( M \). Then it holds

\[
I_{A}(s') \leq \max \left\{ 8 \max_{s'' \in M} I_{s''}(s'), 0 \right\},
\]

i.e. when restricting the optimization to regions of the monotone lower envelope, at least \( \frac{1}{8} \) of the Integrated improvement is obtained.

Proof. Consider the set \( S \) of parent-segments which regions intersect with that of \( s' \) given as

\[
S = \left\{ \text{parent-segment } s \mid [X_1(s), X_2(s)] \cap [X_1(s'), X_2(s')] \neq \emptyset \right\}.
\]

Note that with respect to the children of all other parent-segments the Integrated improvement of \( s'' \) is 0 by definition, since their intersection is empty. We partition the set \( S \) into the subsets \( S_1 \) and \( S_2 \) given as

- \( S_1 := \{ s \in S \mid [X_1(s), X_2(s)] \subseteq [X_1(s'), X_2(s')] \} \),
- \( S_2 := S \setminus S_1 \).

For parent-segments \( s \in S_1 \), we can restrict the segment patch \( s' \) to the interval \([X_1(s), X_2(s)]\) along the \( x \)-axis, since this interval is by definition fully contained in the interval corresponding to \( s' \). By Lemma 4.2, it then holds

\[
\sum_{s'' \in \text{chldn}(s)} I_{s''}(s') \leq 0.
\]

Thus, the contribution of the children of segments in \( S_1 \) to the whole Integrated improvement sum is nonpositive. There are at most two elements in \( S_2 \) since an interval has only two endpoints and the parent-segments are non-overlapping. These parent-segments have at most 4 children each, corresponding to 8 segments in total. Combining the above observations, we get the following decomposition of the sum of Integrated improvement volumes which leads
to the claim

\[ I_A(s') = \sum_{s' \in M} \tilde{I}_v(s') \]

\[ = \sum_{s \in S} \sum_{s' \in \text{chldn}(s)} \tilde{I}_v(s') \]

\[ = \sum_{s \in S_1} \sum_{s' \in \text{chldn}(s)} \tilde{I}_v(s') + \sum_{s \in S_2} \sum_{s' \in \text{chldn}(s)} \tilde{I}_v(s') \]

\[ \leq 0 \]

\[ = \max \left\{ 8 \max_{s \in S_2} \tilde{I}_v(s'), 0 \right\} \]

\[ = \max \left\{ 8 \max_{s' \in M} \tilde{I}_v(s'), 0 \right\}. \]

As we have now shown that optimizing within the regions given by the segments of the monotone lower envelope is sufficient, we can obtain the Integrated improvement guarantee with a corresponding constant factor.

**Theorem 4.2** (Realization of the Integrated improvement guarantee). The Adaptive Patch Approximation Algorithm 2 fulfills the Integrated improvement guarantee of Definition 3.2 with \( \kappa = \frac{1}{8}(1 - \beta) \). Here \( 1 - \beta \) is the approximation factor for finding individual segment patches.

**Proof.** Let \( A \) be the current approximation \( A \) generated by Algorithm 2 and \( M \) the corresponding monotone lower envelope. Let \( s' \) be an arbitrary segment patch. We have to show that the optimal segment patch \( s' \) found during the optimization within a region of \( M \) achieves an Integrated improvement \( I_A(s') \) of at least a fraction \( \kappa = \frac{1}{8}(1 - \beta) \) of the Integrated improvement \( I_A(s) \) of \( s \). By Lemma 4.3 it holds

\[ I_A(s') \leq \max \left\{ 8 \max_{s' \in \text{chldn}(s)} \tilde{I}_v(s'), 0 \right\} \]

In the case that \( I_A(s) = 0 \) the Integral improvement guarantee is trivially fulfilled as our algorithm always uses segment patches with nonnegative integral improvement. We thus can assume \( I_A(s') > 0 \) in the following. By the inequality above there exists a segment \( s' \in M \) in the monotone lower envelope \( M \) which fulfills

\[ \tilde{I}_v(s') \geq \frac{1}{8} I_A(s'). \]

Since we perform an approximate optimization of the integral improvement with respect to the Integrated improvement, we also compute a segment patch \( s \) with (approximately) maximal Integrated improvement w.r.t. the segment \( s' \) of the monotone lower envelope. Since we have an approximation factor of \( 1 - \beta \) for the individual optimization, it holds

\[ I_v(s) \geq (1 - \beta) I_v(s'). \]

Because our algorithms outputs a segment patch \( s' \) with an Integrated improvement of at least
$I_{\varepsilon}(s)$ provided by the alternative patch $s$, we obtain

$$I_A(s') \geq I_{\varepsilon}(s) \geq (1 - \beta)I_{\varepsilon}(s) \geq \frac{1}{8}(1 - \beta)I_A(s)$$

This implies that the Integrated improvement guarantee (Definition 3.2) is fulfilled with the factor $\kappa = \frac{1}{8}(1 - \beta)$.

The combination of this guarantee with our previous convergence analysis based on the Integrated improvement guarantee in Theorem 3.1 establishes that the Adaptive Patch Approximation Algorithm 2 computes approximations to the Pareto frontier with a convergence rate that almost matches the theoretical lower bound provided by the patch complexity.

Main theorem 4.1 (Performance of the Adaptive Patch Approximation Algorithm). The number of iterations of the Adaptive Patch Approximation Algorithm 2 for reaching a difference volume that is less or equal to $\varepsilon$ is bounded in terms of the segment patch complexity $N_{\delta_{vol}}^S$ by

$$\frac{16}{1 - \beta} \bar{\lambda} \left( N_{\delta_{vol}}^S(\varepsilon/2) \right) \ln(1/\varepsilon).$$

The number of MIP optimizations performed by the algorithm is asymptotically bounded by the same quantity since per iteration at most 5 MIPs need to be optimized.

Here $1 - \beta$ is a constant factor given by the approximation factor with respect to optimal Integrated improvement. The function $\bar{\lambda}$ denotes the monotone lower envelope complexity, a function that has almost linear growth rate.

Using the bounds on the monotone lower envelope complexity from Corollary 3.1 we get the alternative upper bounds on the number of iterations

$$64N_{\delta_{vol}}^S(\varepsilon/2) \ln(N_{\delta_{vol}}^S(\varepsilon/2)) \frac{\ln(1/\varepsilon)}{1 - \beta}$$

and

$$1664N_{\delta_{vol}}^S(\varepsilon/2) \left( \alpha(2N_{\delta_{vol}}^S(\varepsilon/2)) + 1 \right) \frac{\ln(1/\varepsilon)}{1 - \beta}.$$

Remark 4.1 (Near-optimality of the algorithm). Main theorem 4.1 shows that the algorithm is asymptotically almost optimal. By the definition of the segment patch complexity $N_{\delta_{vol}}^S$, to reach a difference volume of at most $\varepsilon$ any algorithm needs to output at least $N_{\delta_{vol}}^S(\varepsilon)$ many segment patches. The factor in the algorithm is $N_{\delta_{vol}}^S(\varepsilon/2)$. Although this value can be slightly larger than the lower bound $N_{\delta_{vol}}^S(\varepsilon)$, it is typically only a small constant factor away from it.

The factor $\bar{\lambda} \left( N_{\delta_{vol}}^S(\varepsilon/2) \right)$ is necessary since our algorithm outputs a set of non-overlapping segment patches in the end. The optimal Pareto frontier consists of $N_{\delta_{vol}}^S(\varepsilon)$ many segment patches which can be however overlapping. To achieve this with non-overlapping segments, in the worst case $\bar{\lambda} \left( N_{\delta_{vol}}^S(\varepsilon) \right)$ many segments are needed. Thus our algorithm cannot be improved in this respect as long as it outputs non-overlapping segments patches.
The factor $\ln(1/\varepsilon)$ is coming from the fact that we do a greedy choice in each step that is only approximately optimal. Due to this property, a reduction of the difference volume by a constant factor can only be guaranteed after a number of patches proportional to $N^{S_{\delta_{vol}}}(\varepsilon)$ have been added. Hence, asymptotically at least $N^{S_{\delta_{vol}}}(\varepsilon)\ln(1/\varepsilon)$ many patches need to be added.

5. Numerical evaluation

5.1. Benchmark problems

To complement the strong theoretical convergence rate we evaluate our algorithm on several benchmark problems. In order to obtain stable results, we apply the algorithm on many randomly generated instances of a problem type with same sizes. This allows to reduce deviations caused by single instances. We use the following benchmark sets:

- The classic benchmark problem by Mavrotas and Diakoulaki [11] which is widely used in the literature. The exact specification of this synthetic problem is given in Appendix B. We use the standard version of the problem, where the number of constraints, discrete and continuous variables are in a fixed proportion, thus requiring only a single size parameter.

- A capacitated facility location problem. We use it in a version with two objective functions containing all the variables but with independently generated coefficients. The size parameter describes the number of customers that is chosen equal to the number of possible facility locations. The detailed formulation is described in Appendix C.

- The biobjective assignment benchmark used in [12] containing only integral variables, described in Appendix D. Although our algorithm is not designed for this pure integer problem, the algorithm solves it with acceptable efficiency.

Computation configuration. In our implementation of the algorithm, we choose the sequence $x_0 = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ as the touching points of the tangents of the approximation of the logarithm. The corresponding approximation error $\beta'$ has the value $\beta' = -\ln(1/2) - 1 - \ln(-\ln(1/2)) \approx 0.05966$. The corresponding relative error for optimizing the product is thus $\beta = 1 - \exp(-2\beta') \approx 0.11247$. Our implementation hence fulfils by Theorem 4.2 the Integrated improvement guarantee with $\kappa = \frac{1}{8} (1 - \beta) \approx 0.11094$.

We implemented the algorithms and appropriate data structures in the programming language F#. For solving the mixed-integer programs formulated by our algorithm, we use Gurobi version 8.1 [37]. We choose the default accuracy settings of Gurobi. The benchmarks were run on a personal laptop with an Intel i7-8665U CPU with 1.9 GHz and 16 GB RAM.

To illustrate the convergence of the algorithm, we additionally let the algorithm compute approximate values for the difference volume and the $\varepsilon$-indicator value in each iteration. For estimating the difference volume, we use a Monte-carlo approach, based on solving $\varepsilon$-constraint subproblems at randomly chosen bounds for one of the objectives. For computing the $\varepsilon$-indicator value, we use a method described in [32]. This method allows us to compute the $\varepsilon$-indicator value exactly also for our piecewise-linear approximations. The details of the MIP-formulations employed for the calculation are however out of the scope of this paper. Running times for these evaluations are not included in our following analysis.
5.2. Computational results on mixed-integer problems

The Benchmark problems of Mavrotas and Diakoulaki. We illustrate the convergence of our algorithm by plotting the evolution of approximation quality measures against the number of iterations. We show both the difference volume which forms the basis of our algorithm, and the $\varepsilon$-indicator value as an alternative approximation quality measure for comparison purposes.

Such a log-log plot is shown in Figure 9 for the benchmark problem of Appendix B with 320 variables. The plots show the consistent convergence behaviour of the algorithm for all sizes of the underlying mixed-integer program, as long as the structure stays the same. The difference volume is decreasing consistently with the rate of a polynomial, after some initial iterations. In the log-log plot this is exhibited by an approximately straight line.

![Figure 9](image)

Figure 9. Dependence of the approximation quality on the number of iterations. Average over 10 instances of the benchmark problem by Mavrotas of Appendix B with 320 variables

In Figure 10 the convergence of our algorithm on various instance sizes is illustrated. The figure shows that the number of needed iterations for similar accuracy values only increases slightly with larger instance sizes. The convergence of the difference volume is more consistent than the convergence of the $\varepsilon$-indicator value since the algorithm focuses on the difference volume.

![Figure 10](image)

Figure 10. Heatmap of the approximation quality after a given number of iterations for varying instance sizes of the benchmark of Mavrotas. The color corresponds to the logarithm to the base 10 of the respective approximation quality measure.
Comparison to other algorithms. Since the benchmark problem by Mavrotas and Diakoulaki is widely used in the literature, it is well suited to perform a comparison of various algorithms. We compare our algorithm to results reported by the following papers that also use this benchmark problem:

- [13, Table 8.2],
- [18, Table 3],
- [4, Table 1], using the CT column,
- [3, Table 2 and Table 3], using results from the SIS-version of the algorithm (the version with the best performance on these instances),
- [38, Table 2], with accuracy parameter $\delta = 10^{-2}$.

Although detailed running time results are provided, no clear guarantees of the approximation quality are given in the works above. Parameters that influence the resulting approximation quality are often specific to an algorithm and hard to compare. We report here the results corresponding to medium accuracy settings. However, the relative accuracy between the different algorithms is hard to estimate. For a better comparison, we also restrict the reviewed results to the simple form of the benchmark, with only binary and continuous variables. The used instance sizes are varying, however all papers use the convention that the total number of variables always equals the number of constraints and exactly half of the variables are binary respectively continuous.

In Figure 11, a comparison of the reported running times for various benchmark sizes is given. Since our algorithm produces approximations iteratively with increasing approximation quality, we give the timings until a medium accuracy of a difference volume $\delta_{vol}$ of $10^{-2}$, respectively a high accuracy of a difference volume $\delta_{vol}$ at most $10^{-3}$ is reached, as well as a very high accuracy of $2 \cdot 10^{-4}$ which is close to the numerical limits for the underlying MIP solver. The timings for our algorithms are averages over 5 randomly generated instances each.

The comparison of Figure 11 shows that our algorithm is competitive with the algorithms in the literature. In particular, the asymptotic growth of the running time of our algorithm as a function of the instance size is significantly slower than for other algorithms. This allows our algorithm to compute good approximations for benchmark of double the size than previously considered in the literature (640 variables instead of previously 320) with a reasonable
running time.

The results of [18, Table 13] provide running times measured until an approximation quality equivalent to a difference volume of 0.02 is reached. A comparison shows that our running time for a similar rough approximation (with a difference volume of $10^{-2}$) has an equivalent asymptotic growth rate. For obtaining approximations with a very good quality, e.g. with a difference volume less than $2 \cdot 10^{-4}$, the running time of our algorithm is slightly worse than other approaches in the literature which are tailored for this task of obtaining almost-exact Pareto frontiers.

The comparison of the number of MIPs solved during the run of the algorithms in Figure 12 shows that this number increases in the case of our algorithm only slightly with larger instance sizes. Our number of MIPs to solve grows slower than that of other algorithms because of the faster convergence of our algorithm. Due to the slightly more complicated MIP formulations used by our algorithms, the time used per MIP solution is however larger for our algorithm. This shows the potential provided by the faster convergence of our algorithm: Improvements to the capability of solving the special MIP-formulations of our algorithm can lead to significant improvements on the current state of the art for the total time needed to approximate Pareto frontiers.

![Figure 12. Comparison of the required number of MIP solutions by several algorithms for the benchmark problems by Mavrotas and Diakoulaki [11]. The instance size corresponds to the total number of variables.](image)

**The capacitated facility location benchmark.** As a second benchmark problem we test our algorithm on the capacitated facility location problem. We consider here a bicriteria version where both continuous and discrete variables appear in both objectives, to create a characteristic bicriteria mixed-integer program, as detailed in Appendix C. Figure 13 shows that our algorithm also exhibits a stable convergence for this benchmark. The number of needed iterations only increases slightly with larger instance sizes.

### 5.3. Behaviour on pure integer programs

Although our algorithm was designed for mixed-integer programs with a significant continuous component, it can also be used for multicriteria pure integer programs. The individual steps could be simplified in the case of a pure integer program, since a segment patch in
this case always reduces to a single point. However, the following experiments show that the convergence behaviour is still competitive with other algorithms that are specialized on pure integer programs.

As a pure integer benchmark problem we use a biobjective assignment problem, which is described in detail in Appendix D. Unlike problems with a significant continuous part, the Pareto frontier of instances of the biobjective assignment problem consists only of horizontal segments. Figure 14 shows that our algorithm nevertheless exhibits a satisfactory convergence on this problem.

Since the biobjective assignment benchmark was also used in [12], a numerical comparison is possible. In Figure 15 a comparison of the running times to our algorithm for various instance sizes is given. Although the algorithm in [12] is specialized for solving this type of pure-integer biobjective problems, our algorithm is competitive with this algorithm.

Figure 13. Heatmap of the approximation quality after a given number of iterations for varying instance sizes of the capacitated facility location benchmark. The color corresponds to the logarithm to the base 10 of the respective approximation quality measure.

Figure 14. Heatmap of the approximation quality after a given number of iterations for varying instance sizes of the biobjective assignment benchmark. The color corresponds to the logarithm to the base 10 of the respective approximation quality measure.

6. Conclusion and Future Work

In this paper, we presented the Adaptive Patch Approximation algorithm for approximating bicriteria convex mixed-integer programs based on patches. To our knowledge this is the first
approximation algorithm for this class of problems that has a provable performance bound close to a non-trivial theoretical optimum, namely the segment patch complexity. This raises the question how this intrinsic property of a bicriteria mixed-integer program behaves. Which characteristics of a mixed-integer program are required in order to guarantee certain growth rates of the segment patch complexity? Our numerical results suggest that the segment patch complexity only grows slowly for practically relevant MIPs, however a theoretical study of the underlying reasons is warranted.

The numerical results show that the algorithm is competitive with other state-of-the-art methods. In order to make it even more usable in practice, several enhancements could be made:

- The algorithm could use the feedback provided by a user to dynamically focus on the part of the Pareto frontier that is currently most important to the decision making process.
- The transformed mixed-integer programs could be solved only approximately to obtain a speed-up. However, care has to be taken to ensure that the overall convergence properties of the algorithm are not affected.
- Instead of computing a segment patch given by just two points, patches with a higher number of base points could be used to better capture the curvature of the Pareto frontier.

Another goal is to extend this methodology to the case of optimization problems with three or more criteria. The notions of difference volume and patch complexity are also valid in this general case. However, several questions have to be answered to develop a generalization of the algorithm:

- What is a suitable higher-dimensional analogue of a segment patch?
- How can a convergence rate be shown that is within a small factor of the patch complexity?
- Which growth rate does the patch complexity have in higher dimensions?
Acknowledgments

The author is grateful to Ina Bertz and Karl-Heinz Küfer for providing valuable comments which improved the exposition.

Declaration of interest

The author has no potential conflict of interest.

References

[1] Ruzika S, Wiecek MM. Approximation methods in multiobjective programming. J Optim Theor Appl. 2005;126(3):473–501.
[2] de Santis M, Eichfelder G, Niebling J, et al. Solving multiobjective mixed integer convex optimization problems. Optimization Online. 2020; Available from: http://www.optimization-online.org/DB_HTML/2019/05/7229.html
[3] Perini T, Boland N, Pecin D, et al. A criterion space method for biobjective mixed integer programming: The boxed line method. INFORMS J Comput. 2020;32(1):16–39.
[4] Fattahi A, Turkay M. A one direction search method to find the exact nondominated frontier of biobjective mixed-binary linear programming problems. Eur J Oper Res. 2018;266(2):415–425.
[5] Laumanns M, Thiele L, Zitzler E. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. Eur J Oper Res. 2006;169(3):932–942.
[6] Döchter K, Klamroth K. A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. J Global Optim. 2015;61(4):643–676.
[7] Rote G. The convergence rate of the sandwich algorithm for approximating convex functions. Computing. 1992;48(3-4):337–361.
[8] Gruber PM, Kenderov P. Approximation of convex bodies by polytopes. Rend Circ Mat Palermo. 1982;31(2):195–225.
[9] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput. 1999;3(4):257–271.
[10] Faulkenberg SL, Wiecek MM. On the quality of discrete representations in multiple objective programming. Optim Eng. 2010;11(3):423–440.
[11] Mavrotas G, Diakoulaki D. A branch and bound algorithm for mixed zero-one multiple objective linear programming. Eur J Oper Res. 1998;107(3):530–541.
[12] Stidsen T, Andersen KA, Dammann B. A branch and bound algorithm for a class of biobjective mixed integer programs. Manage Sci. 2014;60(4):1009–1032.
[13] Belotti P, Soylu B, Wiecek MM. Fathoming rules for biobjective mixed integer linear programs: Review and extensions. Discrete Optim. 2016;22:341–363.
[14] Cacchiani V, D’Ambrosio C. A branch-and-bound based heuristic algorithm for convex multi-objective minlps. Eur J Oper Res. 2017;260(3):920–933.
[15] Vassilvitskii S, Yannakakis M. Efficiently computing succinct trade-off curves. Theor Comput Sci. 2005;348(2-3):334–356.
[16] Kim IY, de Weck OL. Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation. Struct Multidiscip Optim. 2006;31(2):105–116.
[17] Eichfelder G. Scalarizations for adaptively solving multi-objective optimization problems. Comput Optim Appl. 2009;44(2):249–273.
[18] Boland N, Charkhgard H, Savelsbergh M. A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method. INFORMS J Comput. 2015;27(4):597–618.
[19] Reuter H. An approximation method for the efficiency set of multiobjective programming problems. Optim. 1990;21(6):905–911.
Appendix A. Construction of linear inequalities to approximate the logarithm

In this appendix we provide a proof for Lemma 4.1 by constructing a sequence of tangent points which yield a sufficient approximation of the logarithm.
Construction of the tangent points. We use tangents for the approximation at some positions \(x_1, \ldots, x_k \in [0, 1]\) which we need to find. The goal is to choose these positions such that the approximated value is always larger than \(\ln(x)\), with an additive difference of at most \(\beta'\).

Thus, at the position \(x = 1\), the approximation should have a value of at most \(\ln(1) + \beta' = \beta'\).

To minimize the number of line segments needed for the approximation, we should choose the value as large as allowed, i.e. the approximation at \(x = 1\) should be exactly equal to \(\beta'\).

Let \(x_0 \in [0, 1]\) be the touching point of the tangent. Since the tangent should have a value of \(\beta'\) at \(x = 1\), we get the equality

\[
\ln(x_0) + \frac{1 - x_0}{x_0} = \beta',
\]

where we used \(\frac{d}{dx} \ln(x) = \frac{1}{x}\). This equality leads to, using the condition \(x_0 \in [0, 1], \beta' \geq 0\):

\[
\ln(x_0) + \frac{1}{x_0} = \beta' + 1
\]

\[
\iff -\frac{1}{x_0} \exp\left( -\frac{1}{x_0} \right) = -\exp(-\beta' - 1)
\]

\[
\iff -\frac{1}{x_0} = W_{-1}(\exp(-\beta' - 1))
\]

\[
\iff x_0 = -\frac{1}{W_{-1}(\exp(-\beta' - 1))},
\]

where \(W_{-1}\) is the non-principal branch of the Lambert function, i.e. for \(-1/e \leq z \leq 0\), the value \(W_{-1}(z)\) is the unique solution \(w \leq -1\) of \(w \exp(w) = z\). The survey [39] provides an overview on the Lambert function and its various applications.

We now search for the next position \(z_1\) where an additive error of \(\beta'\) of the tangent through \(x_0\) to the function \(\ln(x)\) occurs. This position \(z_1\) is characterized by the equation

\[
\ln(x_0) + (z_1 - x_0) \cdot \frac{1}{x_0} - \ln(z_1) = \beta',
\]

which can be equivalently transformed, using the substitution \(z' = \frac{z}{x_0}\), to

\[
z' - \ln(x_0z') = \beta' + 1 - \ln(x_0)
\]

\[
\iff z' - \ln(z') = \beta' + 1
\]

\[
\iff z' = -W(\exp(-\beta' - 1))
\]

\[
\Rightarrow z_1 = x_0z' = \frac{W_0(-\exp(-\beta' - 1))}{W_{-1}(\exp(-\beta' - 1))},
\]

where we used the principal branch \(W_0\) of the Lambert function to get a solution \(z' < 1\). Otherwise we would have obtained the solution \(z' = 1\) which is obvious from the construction of \(x_0\). The principal branch \(W_0(z)\) is defined for values \(z \geq -1\) as the unique solution \(w \geq -1\) of \(w \exp(w) = z\).

Recursive application. We now show that values for the other breakpoints scale in the same way.
Consider $x_1$, the touching position of the tangent such that the difference to the natural logarithm at position $z_1$ is $\beta'$, which is characterized by the equation

$$\ln(x_1) + (z_1 - x_1) \cdot \frac{1}{x_1} - \ln(z_1) = \beta'.$$

Using the substitution $x'_1 = \frac{z_1}{x_1}$, we obtain the equivalent equation

$$\ln(x'_1) + \frac{1}{x'_1} = \beta' + 1$$

which has the exact same form as the equation for $x_0$, implying $x'_1 = x_1$ and by resubstitution $x_1 = x'_1 \cdot z_1 = x_0 z_1$. We apply the analogous analysis for the position $z_2$ at which the difference of the tangent touching at $x_1$ has a difference of $\beta'$ to the function $\ln$. This leads to the equation

$$\ln(x_1) + (z_2 - x_1) \cdot \frac{1}{x_1} - \ln(z_2) = \beta',$$

which can be transformed via the substitution $z'_2 = \frac{z_2}{z_1}$ and using the result $x_1 = z_1 \cdot x_0$ to

$$\ln(x_0) + (z'_2 - x_0) \cdot \frac{1}{x_0} - \ln(z'_2) = \beta'.$$

Since this matches again the defining equation for $z_1$, we have $z'_2 = z_1$ and by resubstitution $z_2 = z'_2 \cdot z_1 = z_1^2$. For the next positions $z_3, z_4, \ldots$ it follows analogously that $z_k = x'_1$ for all $k \in \mathbb{N}$.

In order to get an additive approximation with an error at most $\beta'$ on the interval $[\delta_0, 1]$, we only need to choose the $k$ tangents at the points $x_0, \ldots, x_{k-1}$ with the number of points $k$ chosen such that $x'_1 \leq \delta_0$ since this implies that only for values less than $\delta_0$ an additive error bigger than $\beta'$ can occur. Solving this for $k$ leads to

$$k = \lceil \log_{x'_1} (\delta_0) \rceil = \left\lceil \frac{\ln(\delta_0)}{\ln(z_1)} \right\rceil = \left\lceil \frac{\ln(1/\delta_0)}{\ln(1/z_1)} \right\rceil,$$

where $\lceil \cdot \rceil$ is the ceiling function. This shows that the dependence with respect to $\delta_0$ is logarithmic which implies that even for values of $\delta_0$ very close to 0 we only need a relative small number of tangents.

**Bounding the step length.** We now bound the value of $z_1$ from above in terms of $\beta'$. This allows us to bound the number of tangents required to reach a certain approximation bound. For this we use the defining equation

$$z_1 = \frac{W_0(-\exp(-\beta' - 1))}{W_{-1}(-\exp(-\beta' - 1))}.$$  

Since for all values $\beta' \geq 0$ the values $W_0(-\exp(-\beta' - 1)) < 0$ and $W_{-1}(-\exp(-\beta' - 1)) < 0$ are negative, we need a lower bound for $W_0$ and an upper bound for $W_{-1}$. We use the following
upper bound on $W_{-1}$ for all $\beta' \geq 0$ shown in [40]

$$W_{-1}(-\exp(-\beta' - 1)) \leq -1 - \frac{2}{3} \beta' \leq -1 - \sqrt{2\beta'}.$$  

By inserting the trivial bound $W_0(x) \geq -1$ for all $x \geq -1/e$, we obtain

$$z_1 = \frac{W_0(-\exp(-\beta' - 1))}{W_{-1}(-\exp(-\beta' - 1))} \leq \frac{-1}{1 - \sqrt{2\beta'}} = \frac{1}{1 + \sqrt{2\beta'}}.$$  

Finally, this yields the desired bound on the number of needed tangents:

$$k = \left\lceil \frac{\ln(1/\delta_0)}{\ln(1/z_1)} \right\rceil \leq \left\lceil \frac{\ln(1/\delta_0)}{\ln(1 + \sqrt{2\beta'})} \right\rceil,$$

as $\delta_0 > 0$, exhibiting the asymptotic dependence of the number of needed linear inequalities for an additive accuracy $\beta'$ close to 0.

**Appendix B. Benchmark problem of Mavrotas and Diakoulaki**

The benchmark multicriteria MIP proposed in [11] is widely used in the literature for numerical studies. It is a randomly generated synthetic problem. We follow here roughly the presentation in [38] and restrict it to the case of two criteria to simplify notation. Additionally, we change the optimization direction to minimization. The problem has the following form, where $x$ and $y$ are the decision variables:

$$\min \left( \sum_{i=1}^{n} c_i^{(1)} x_i + \sum_{j=1}^{q} f_j^{(1)} y_j, \sum_{i=1}^{n} c_i^{(2)} x_i + \sum_{j=1}^{q} f_j^{(2)} y_j \right)$$

s.t. $\sum_{i=1}^{n} a_{ij} x_i + a'_{j} y_j \leq b_j$ for all $j = 1, \ldots, q$

$\sum_{i=1}^{n} a_{ij} x_i \leq b_j$ for all $j = q + 1, \ldots, m - 1$

$\sum_{j=1}^{q} y_j \leq \frac{q}{3}$

$x_i \in \mathbb{R}_{\geq 0}$ for all $i = 1, \ldots, n$

$y_j \in \{0, 1\}$ for all $j = 1, \ldots, q$

The parameter $m$ denotes the total number of constraints, the number of continuous variables is denoted by $n$ and the number of binary variables is given by $q$. We randomly generate the coefficients independently according to the following uniform distributions:

- $c_i^{(1)}, c_i^{(2)} \sim \mathcal{U}([-10, 10])$, $i = 1, \ldots, n$
- $f_j^{(1)}, f_j^{(2)} \sim \mathcal{U}([-200, 200])$, $j = 1, \ldots, q$
- $b_j \sim \mathcal{U}([-1, 20])$, $j = 1, \ldots, m - 1$
- $a_{ij}, a'_{j} \sim \mathcal{U}([-1, 20]) \circ B(0.25)$

36
where \( \mathcal{N}([-1,20]) \circ \mathcal{B}(0.25) \) denotes the distribution with values having a value of 0 with a probability of 0.75 and otherwise uniform values in \([-1,20]\).

Appendix C. Capacitated facility location problem

The capacitated facility location problem consists of a set \( I \) of \( n = |I| \) many possible facility locations that should serve a set \( J \) of customers (here chosen to also have cardinality \( |J| = n \)). A corresponding cost occurs for each opened facility as well as each unit transported from a facility to a customer. We use two independent sets of cost coefficients for the two objective functions, each including all decision variables. The bicriteria capacitated facility location problem is formally given as:

\[
\min \left( \sum_{i \in I} \sum_{j \in J} c_{ij}^{(1)} y_{ij} + \sum_{i=1}^{n} f_i^{(1)} x_i, \sum_{i \in I} \sum_{j \in J} c_{ij}^{(2)} y_{ij} + \sum_{i=1}^{n} f_i^{(2)} x_i \right)
\]

s. t.  
\[ \sum_{i=1}^{n} y_{ij} = 1 \quad \text{for all } j \in J \]
\[ \sum_{j \in J} y_{ij} \leq u_i x_i \quad \text{for all } i \in I \]
\[ y_{ij} \in \mathbb{R}_{\geq 0} \quad \text{for all } i, j \in I \]
\[ x_i \in \{0, 1\} \quad \text{for all } i \in I \]

Here, the coefficients \( c_{ij}^{(1)}, c_{ij}^{(2)} \) represent the transportation cost per unit, \( f_i^{(1)}, f_i^{(2)} \) the cost for opening a facility and \( u_i \) the maximal capacity of a facility. The variable \( x_i \) indicates whether facility \( i \in I \) is opened and \( y_{ij} \) describes the amount sent from facility \( i \) to customer \( j \). We generate the coefficients randomly as follows, all independently from each other:

- We sample the transportation cost coefficients \( c_{ij}^{(1)}, c_{ij}^{(2)} \), \( i \in I, j \in J \) uniformly from the interval \([1,2]\).
- We sample the facility opening costs \( f_i^{(1)}, f_i^{(2)} \), \( i \in I \) uniformly from the interval \([0,10]\).
- The capacity values \( u_i \), \( i \in I \) are sampled uniformly from the interval \([1,10]\).

Appendix D. Biobjective Assignment problem

The biobjective assignment problem asks to find an assignment between two sets. There are weights for each possible chosen assignment. The goal is to minimize this total weight. For each of the objectives, the weights are different. We use the random generation as described...
Formally, the problem can be given as:

\[
\min \left( \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^{(1)} , \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^{(2)} \right)
\]

s. t. \( \sum_{i=1}^{n} x_{ij} = 1 \) for all \( j = 1, \ldots, n \)

\( \sum_{j=1}^{n} x_{ij} = 1 \) for all \( i = 1, \ldots, n \)

\( x_{ij} \in \{0, 1\} \) for all \( i, j = 1, \ldots, n \)

We choose the objective function coefficients \( c_{ij}^{(1)} \) and \( c_{ij}^{(2)} \) randomly independently from the uniform distribution \( \mathcal{U}([0,20]) \).