



An approximation algorithm for a general class of parametric optimization problems

Cristina Bazgan¹ · Arne Herzel²  · Stefan Ruzika² · Clemens Thielen³ · Daniel Vanderpooten¹

Published online: 22 September 2020
© The Author(s) 2020

Abstract

In a (linear) parametric optimization problem, the objective value of each feasible solution is an affine function of a real-valued parameter and one is interested in computing a solution for each possible value of the parameter. For many important parametric optimization problems including the parametric versions of the shortest path problem, the assignment problem, and the minimum cost flow problem, however, the piecewise linear function mapping the parameter to the optimal objective value of the corresponding non-parametric instance (the *optimal value function*) can have super-polynomially many breakpoints (points of slope change). This implies that any optimal algorithm for such a problem must output a super-polynomial number of solutions. We provide a method for lifting approximation algorithms for non-parametric optimization problems to their parametric counterparts that is applicable to a general class of parametric optimization problems. The approximation guarantee achieved by this method for a parametric problem is arbitrarily close to the approximation guarantee of the algorithm for the corresponding non-parametric problem. It outputs polynomially many solutions and has polynomial running time if the non-parametric algorithm has polynomial running time. In the case that the non-parametric problem can be solved exactly in polynomial time or that an FPTAS is available, the method yields an FPTAS. In particular, under mild assumptions, we obtain the first parametric FPTAS for each of the specific problems mentioned above and a $(3/2 + \varepsilon)$ -approximation algorithm for the parametric metric traveling salesman problem. Moreover, we describe a post-processing procedure that, if the non-parametric problem can be solved exactly in polynomial time, further decreases the number of returned solutions such that the method outputs at most twice as many solutions as needed at minimum for achieving the desired approximation guarantee.

This work was supported by the bilateral cooperation project “Approximation methods for multiobjective optimization problems” funded by the German Academic Exchange Service (DAAD, Project-ID 57388848) and by Campus France, PHC PROCOPE 2018 (Project No. 40407WF) as well as by the DFG Grants TH 1852/4-1 and RU 1524/6-1. An earlier version of this work appeared in Bazgan et al. (2019).

Extended author information available on the last page of the article

Keywords Parametric optimization · Approximation algorithm · Parametric assignment problem · Parametric minimum cost flow problem · Parametric shortest path problem · Parametric metric TSP

1 Introduction

In a linear parametric optimization problem, the objective function value of a feasible solution does not only depend on the solution itself but also on a parameter $\lambda \in \mathbb{R}$, where this dependence is given by an affine-linear function of λ . The goal is to find an optimal solution for each possible parameter value, where, under some assumptions (e.g., if the set of feasible solutions is finite), an *optimal* solution can be given by a finite collection of intervals $(-\infty, \lambda_1]$, $[\lambda_1, \lambda_2]$, \dots , $[\lambda_{K-1}, \lambda_K]$, $[\lambda_K, +\infty)$ together with one feasible solution for each interval that is optimal for all values of λ within the corresponding interval.

The function mapping each parameter value $\lambda \in \mathbb{R}$ to the optimal objective value of the non-parametric problem induced by λ is called the *optimal value function* (or the *optimal cost curve*). The above structure of optimal solutions implies that the optimal value function is piecewise linear and concave in the case of a minimization problem (convex in case of a maximization problem) and its *breakpoints* (points of slope change) are exactly the points $\lambda_1, \dots, \lambda_K$ (assuming that K was chosen as small as possible).

There is a large body of literature that considers linear parametric optimization problems in which the objective values of feasible solutions are affine-linear functions of a real-valued parameter. These problems often arise naturally from important non-parametric problems. Karp and Orlin (1981) observe that the minimum mean cycle problem can be reduced to the parametric shortest path problem (Carstensen 1983b; Mulmuley and Shah 2001). Young et al. (2006) note that parametric shortest path problems appear in the process of solving the minimum balance problem, the minimum concave-cost dynamic network flow problem (Graves and Orlin 1985), and matrix scaling (Orlin and Rothblum 1985; Schneider and Schneider 1991). Other prominent examples include the parametric assignment problem (Gassner and Klinz 2010) and the parametric minimum cost flow problem (Carstensen 1983a). Moreover, parametric versions of general linear programs, mixed integer programs, and nonlinear programs (where the most general cases consider also non-affine dependence on the parameter as well as constraints depending on the parameter) are widely studied—see Mitsos and Barton (2009) for an extensive literature review.

The number of breakpoints is a natural measure for the complexity of a parametric optimization problem since it determines the number of different solutions that are needed in order to solve the parametric problem to optimality. Moreover, any instance of a parametric optimization problem with K breakpoints in the optimal value function can be solved by using a general method of Eisner and Severance (1976), which requires to solve $\mathcal{O}(K)$ non-parametric problems for fixed values of the parameter.

Carstensen (1983a) shows that the number of breakpoints in the optimal value function of any parametric binary integer program becomes linear in the number of variables when the slopes and/or intercepts of the affine-linear functions are integers

in $\{-M, \dots, M\}$ for some constant $M \in \mathbb{N}$. In most parametric problems, however, the number of possible slopes and intercepts is exponential and/or the variables are not binary. While there exist some parametric optimization problems such as the parametric minimum spanning tree problem (Fernández-Baca et al. 1996) or several special cases of the parametric maximum flow problem (Arai et al. 1993; Gallo et al. 1989; McCormick 1999; Scutellà 2007) for which the number of breakpoints is polynomial in the input size even without any additional assumptions, the optimal value function of most parametric optimization problems can have super-polynomially many breakpoints in the worst case—see, e.g., Carstensen (1983b), Nikolova et al. (2006) for the parametric shortest path problem, Gassner and Klinz (2010) for the parametric assignment problem, and Ruhe (1988) for the parametric minimum cost flow problem. This, in particular, implies that there cannot exist any polynomial-time algorithm for these problems even if $P = NP$, which provides a strong motivation for the design of approximation algorithms.

To the best of our knowledge, the only existing approximation algorithms for a parametric optimization problem are the approximation schemes for the parametric version of the 0–1 knapsack problem presented in Giudici et al. (2017), Holzhauser and Krumke (2017). Additionally, an approximation scheme for the variant of the 0–1 knapsack problem in which the weights of the items (instead of the profits) depend on the parameter has recently been provided in Halman et al. (2018).

If only positive values are allowed for the parameter λ , linear parametric optimization is closely related to computing supported solutions in biobjective linear optimization problems. Approximating the set of supported solutions in multiobjective optimization has been considered in Daskalakis et al. (2016), Diakonikolas (2011), Diakonikolas and Yannakakis (2008). Under some additional assumptions, approximation in linear parametric optimization problems and approximating supported solutions in biobjective linear optimization problems are equivalent (Diakonikolas 2011; Diakonikolas and Yannakakis 2008). For these overlapping cases, many of the algorithms presented here are similar to the ones studied in Daskalakis et al. (2016), Diakonikolas (2011), Diakonikolas and Yannakakis (2008).

Our contribution In Sect. 3, we provide a polynomial-time (parametric) approximation method for a general class of parametric optimization problems whose non-parametric versions have a polynomial-time approximation algorithm available. This means that, for any problem from this class, the method computes a set of solutions that approximate all feasible solutions for all values of λ within the given interval of allowed parameter values. The approximation guarantee of this method depends on the approximation guarantee achievable for the corresponding non-parametric problem. If the non-parametric problem can be approximated within a factor of $\beta \geq 1$, then, for any $\varepsilon > 0$, the parametric problem can be approximated within a factor of $(1 + \varepsilon) \cdot \beta$ in time polynomial in the size of the input and in $\frac{1}{\varepsilon}$. Thus, if an FPTAS or an exact algorithm for the non-parametric problem is available, we obtain a (parametric) FPTAS for the parametric problem.

Moreover, in Sect. 4, we provide an algorithm for parametric problems whose non-parametric version is solvable exactly that, given any β -approximation S of cardinality $|S|$ for the parametric problem, computes a minimum-cardinality subset S' of S

that still forms a β -approximation. The running time of this algorithm is quadratic in $|S|$ in general. However, we show that, if this algorithm is applied to a β -approximation computed by the approximation method from Sect. 3, then (i) the running time can be reduced to being linear in $|S|$ and (ii) the cardinality of the computed subset S' is at most twice as large as the cardinality of any β -approximation for the given problem instance. We will see that this factor of two is best possible in the sense that, for the studied problem class, no algorithm can achieve a better factor in general.

Our approximation method can be viewed as an approximate version of the well-known Eisner-Severance method for parametric optimization problems (Eisner and Severance 1976). It applies, in particular, to all parametric optimization problems for which the parameter varies on the nonnegative real line, the non-parametric problem can be approximated in polynomial time, and the slopes and intercepts of the value functions of the feasible solutions are nonnegative integer values below a polynomial-time computable upper bound. Under mild assumptions, we obtain the first parametric FPTAS for the parametric versions of the shortest path problem, the assignment problem, and a general class of mixed integer linear programming problems over integral polytopes, which includes the minimum cost flow problem as a special case. As we discuss in the applications of our method presented in Sect. 5, the number of breakpoints can be super-polynomial for each of these parametric problems even under our assumptions, which implies that the problems do not admit any polynomial-time exact algorithms. We moreover obtain a $(\frac{3}{2} + \varepsilon)$ -approximation for the parametric version of the APX-hard metric traveling salesman problem.

This work extends our conference paper (Bazgan et al. 2019), in which a less adaptive version of the algorithm is presented only for the special case where the non-parametric problem can be solved exactly. Moreover, the question of computing approximations with small cardinality is not considered in the conference paper.

2 Preliminaries

In the following, we consider a general parametric minimization or maximization problem Π of the following form:

$$\begin{aligned} \min / \max \quad & f_x(\lambda) := a(x) + \lambda \cdot b(x) \\ \text{s. t. } & x \in X \end{aligned} \quad (1)$$

The goal is to provide an optimal solution to this optimization problem for every $\lambda \in I$, where $I = [\lambda_{\min}, \infty)$ for some $\lambda_{\min} \in \mathbb{R}$, which is given as part of the input of the problem. More precisely, a solution $x_\lambda^* \in X$ is sought, for every $\lambda \in I$, such that

$$f_{x_\lambda^*}(\lambda) = f(\lambda) := \min / \max_{x \in X} f_x(\lambda),$$

where $f : I \rightarrow \mathbb{R}$ is the optimal cost curve mapping each parameter value to its corresponding optimal objective function value. It is assumed that the optimal cost curve is well-defined, i.e., $\min / \max_{x \in X} f_x(\lambda)$ exists for every $\lambda \in I$. It is easy to see that the optimal cost curve f is continuous and concave (for minimization problems) or convex (for maximization problems).

For any $\lambda \in I$, we call the non-parametric problem obtained from Π by fixing the parameter value to λ the *non-parametric version of Π* .

Even though all our results apply to minimization as well as maximization problems, we focus on minimization problems in the rest of the paper in order to simplify the exposition. All our arguments can be straightforwardly adapted to maximization problems.

We assume that the functions $a, b : X \rightarrow \mathbb{R}$ defining the intercepts and slopes of the value functions, respectively, are polynomial-time computable, and that we can compute positive rational bounds LB and UB such that $b(x), f_x(\lambda_{\min}) \in \{0\} \cup [LB, UB]$ for all $x \in X$ in polynomial time. In particular, this implies that LB and UB are of polynomial encoding length.¹ These assumptions ensure that the objective function values $f_x(\lambda)$ are nonnegative for all $x \in X$ and $\lambda \in I$, which is a usual requirement when considering approximation. The bounds LB and UB are guaranteed to exist, e.g., for combinatorial optimization problems for which the decision version of the non-parametric problem is in NP, and also for problems where $a(x)$ and $b(x)$ only attain nonnegative integer values of polynomial encoding length (we can choose $LB = 1$ in this case).

We say that a solution $x \in S \subseteq X$ is *supported in S* if x is optimal within S for some parameter value, i.e., if $f_x(\lambda) = \min_{x' \in S} f_{x'}(\lambda)$ for at least one $\lambda \in I$. If a solution $x \in X$ is supported in X , we simply call x *supported*.

For two solutions $x, x' \in X$ for which $b(x) \neq b(x')$, we define $\mu(x, x') \in \mathbb{R}$ to be the value where the two lines $f_x(\cdot)$ and $f_{x'}(\cdot)$ intersect, i.e., the parameter value for which $f_x(\mu(x, x')) = f_{x'}(\mu(x, x'))$ (which is not necessarily contained in I):

$$\mu(x, x') := \frac{a(x') - a(x)}{b(x) - b(x')}$$

Note that $\mu(x, x') = \mu(x', x)$ for all $x, x' \in X$ with $b(x) \neq b(x')$. The following simple lemma covers the most important property of $\mu(x, x')$.

Lemma 1 *If $b(x) > b(x')$, then the following two equivalences hold:*

$$f_x(\lambda) \leq f_{x'}(\lambda) \Leftrightarrow \lambda \leq \mu(x, x')$$

and

$$f_x(\lambda) \geq f_{x'}(\lambda) \Leftrightarrow \lambda \geq \mu(x, x').$$

Proof $f_x(\lambda)$ can be written as

$$\begin{aligned} f_x(\lambda) &= f_x(\mu(x, x')) + (\lambda - \mu(x, x')) \cdot b(x) \\ &= f_{x'}(\mu(x, x')) + (\lambda - \mu(x, x')) \cdot b(x) \end{aligned}$$

for any $x, x' \in X$ and $\lambda \in I$. □

¹ Note that, the *numerical values* of LB and UB can still be *exponential* in the input size of the problem.

Our goal is approximating linear parametric optimization problems, that is, for every $\lambda \in I$, finding a solution $x_\lambda \in X$ that is approximately optimal for λ . We want the total number of solutions used for approximating the parametric problem to be finite. When given a finite set $S \subseteq X$ of solutions, we would obviously assign them to the parameter values in a way such that each parameter value $\lambda \in I$ is assigned a solution $x_\lambda \in S$ such that

$$f_{x_\lambda} = \min_{x \in S} f_x(\lambda). \quad (2)$$

This can be done easily using Lemma 1. A simple (yet not necessarily the most efficient) way to do this is to compute the value $\mu(x, x')$ for every pair of solutions $x, x' \in X$ with $b(x) \neq b(x')$, partition I into subintervals defined by these values, and then determine a solution x_λ satisfying (2) for an arbitrary parameter value λ in each of these subintervals. Lemma 1 then ensures that x_λ also satisfies (2) for all other parameter values inside the corresponding interval. It is, therefore, reasonable to define approximations for parametric optimization problems as follows:

Definition 1 For $\alpha \geq 1$, an α -approximation for an instance of a parametric optimization problem Π is a finite set $S \subseteq X$ such that, for any $\lambda \in I$, there exists a solution $x \in S$ that is α -approximate for (the non-parametric problem instance obtained by fixing the parameter value to) λ , i.e., for which

$$f_x(\lambda) \leq \alpha \cdot f(\lambda).$$

An algorithm A that computes an α -approximation in polynomial time for every instance of Π is called an α -approximation algorithm for Π .

A polynomial-time approximation scheme (PTAS) for Π is a family $(A_\varepsilon)_{\varepsilon > 0}$ of algorithms such that, for every $\varepsilon > 0$, algorithm A_ε is a $(1 + \varepsilon)$ -approximation algorithm for Π . A PTAS $(A_\varepsilon)_{\varepsilon > 0}$ for Π is called a fully polynomial-time approximation scheme (FPTAS) if the running time of A_ε is additionally polynomial in $1/\varepsilon$.

3 A general approximation method

We now present our method for approximating a general parametric optimization problem Π as in (1). We assume that there is an algorithm available that can approximate the non-parametric version of Π for any $\lambda \in I$. More precisely, we assume that, for some $\beta \geq 1$, we have an algorithm ALG_β that, given any fixed $\lambda \in I$, computes a β -approximate solution for λ .² Our method uses ALG_β as a blackbox in order to compute an approximation for the parametric problem. The approximation quality achievable by the method is arbitrarily close to β , i.e., for any $\varepsilon > 0$, we can compute a $((1 + \varepsilon) \cdot \beta)$ -approximation for the parametric problem. If the non-parametric problem can be solved exactly in polynomial time (i.e., if $\beta = 1$) or admits a (non-parametric) FPTAS, our method yields a (parametric) FPTAS.

² The approximation guarantee β may be a constant or depend on the given instance, but must be independent of λ .

The running time of ALG_β will be denoted by T_{ALG_β} , where we assume that this running time is at least as large as the time needed to compute the objective value $f_x(\lambda) = a(x) + \lambda \cdot b(x)$ of any feasible solution $x \in X$ in the non-parametric problem.³

We first describe the general functioning of the algorithm before formally stating it and proving several auxiliary results needed for proving its correctness and analyzing its running time.

The algorithm starts with the initial parameter interval $[\lambda_*, \lambda^*]$, where λ_* is chosen such that a β -approximate solution x_* for λ_* is $((1 + \varepsilon) \cdot \beta)$ -approximate for all $\lambda \in [\lambda_{\min}, \lambda_*]$ and λ^* is chosen such that a β -approximate solution x^* for λ^* is $((1 + \varepsilon) \cdot \beta)$ -approximate for all $\lambda \in [\lambda^*, +\infty)$ (see Proposition 1).

The algorithm maintains a queue Q , whose elements $([\lambda_\ell, \lambda_r], x_\ell, x_r)$ consist of a subinterval $[\lambda_\ell, \lambda_r]$ of the interval $[\lambda_*, \lambda^*]$ and β -approximate solutions x_ℓ, x_r for the respective interval boundaries λ_ℓ, λ_r . The queue is initialized as $Q = \{([\lambda_*, \lambda^*], x_*, x^*)\}$, where x_*, x^* are β -approximate for λ_*, λ^* , respectively.

After the initialization, in each iteration, the algorithm extracts an element $([\lambda_\ell, \lambda_r], x_\ell, x_r)$ from the queue and first checks if one of the two conditions $f_{x_\ell}(\lambda_r) \leq f_{x_r}(\lambda_r)$ or $f_{x_r}(\lambda_\ell) \leq f_{x_\ell}(\lambda_\ell)$ holds, in which case one of the two solutions x_ℓ, x_r is β -approximate for the whole interval $[\lambda_\ell, \lambda_r]$. Otherwise, the algorithm considers the line connecting $(\lambda_\ell, f_{x_\ell}(\lambda_\ell))$ and $(\lambda_r, f_{x_r}(\lambda_r))$. If this line is not more than a factor of $(1 + \varepsilon)$ (measured in the second coordinate) away from either the line f_{x_ℓ} or the line f_{x_r} , we know that x_ℓ or x_r is $((1 + \varepsilon) \cdot \beta)$ -approximate within the whole interval $[\lambda_\ell, \lambda_r]$ (see Proposition 2). This condition can be checked easily at $\lambda = \mu(x_\ell, x_r)$, where f_{x_ℓ} and f_{x_r} intersect, since this must be the value within $[\lambda_\ell, \lambda_r]$ where this distance is maximum.

Otherwise, $[\lambda_\ell, \lambda_r]$ is bisected into the subintervals $[\lambda_\ell, \lambda_m]$ and $[\lambda_m, \lambda_r]$, where $\lambda_m := \mu(x_\ell, x_r)$. This means that a β -approximate solution x_m for λ_m is computed and the two triples $([\lambda_\ell, \lambda_m], x_\ell, x_m)$ and $([\lambda_m, \lambda_r], x_m, x_r)$ are added to the queue in order to be explored. Proposition 3 states that $[\lambda_\ell, \lambda_r]$ will not be bisected if λ_m is too close to either λ_ℓ or λ_r , which ensures a polynomial number of iterations performed by the algorithm in total.

Before proving the auxiliary results mentioned in the algorithm description above, we state two helpful lemmas about general properties of approximations. The first lemma shows that a solution that is α -approximate for two different parameter values must also be α -approximate for all parameter values in between.

Lemma 2 *Let $\emptyset \neq [\underline{\lambda}, \bar{\lambda}] \subseteq I$ be a compact interval and let $\alpha \geq 1$. If some solution $x \in X$ is α -approximate for $\underline{\lambda}$ and for $\bar{\lambda}$ then x is α -approximate for all $\lambda \in [\underline{\lambda}, \bar{\lambda}]$.*

³ This technical assumption—which is satisfied for most relevant algorithms—is made in order to be able to express the running time of our algorithm in terms of T_{ALG_β} . If the assumption is removed, our results still hold when replacing T_{ALG_β} in the running time of our algorithm by the maximum of T_{ALG_β} and the time needed for computing any value $f_x(\lambda)$

Algorithm 1: An approximation algorithm for parametric optimization problems

```

input : An instance of a parametric optimization problem  $\Pi$  as in (1),  $\varepsilon > 0$ , a  $\beta$ -approximation
        algorithm  $\text{ALG}_\beta$  for the non-parametric version of  $\Pi$ 
output: A  $((1 + \varepsilon) \cdot \beta)$ -approximation for the given instance of  $\Pi$ 

1 Compute LB and UB
2  $\lambda_* \leftarrow \lambda_{\min} + \frac{1}{\beta} \cdot \varepsilon \cdot \frac{\text{LB}}{\text{UB}}$ ;  $\lambda^* \leftarrow \lambda_{\min} + \beta \cdot \frac{1}{\varepsilon} \cdot \frac{\text{UB}}{\text{LB}}$ 
3  $x_* \leftarrow \text{ALG}_\beta(\lambda_*)$ ;  $x^* \leftarrow \text{ALG}_\beta(\lambda^*)$ 
4  $Q \leftarrow \{([\lambda_*, \lambda^*], x_*, x^*)\}$  /* queue of intervals still to be considered */
5  $S \leftarrow \{x_*, x^*\}$  /* solution set */
6 while  $Q \neq \emptyset$  do
7   Extract some element  $([\lambda_\ell, \lambda_r], x_\ell, x_r)$  from  $Q$ 
8   if  $f_{x_\ell}(\lambda_r) \leq f_{x_r}(\lambda_r)$  then
9      $S \leftarrow S \cup \{x_\ell\}$ 
10  else if  $f_{x_r}(\lambda_\ell) \leq f_{x_\ell}(\lambda_\ell)$  then
11     $S \leftarrow S \cup \{x_r\}$ 
12  else
13     $\lambda_m \leftarrow \mu(x_\ell, x_r)$ 
14    if  $f_{x_\ell}(\lambda_m) \leq (1 + \varepsilon) \cdot \left( \frac{\lambda_r - \lambda_m}{\lambda_r - \lambda_\ell} \cdot f_{x_\ell}(\lambda_\ell) + \frac{\lambda_m - \lambda_\ell}{\lambda_r - \lambda_\ell} \cdot f_{x_r}(\lambda_r) \right)$  then
15       $S \leftarrow S \cup \{x_\ell, x_r\}$ 
16    else
17       $x_m \leftarrow \text{ALG}_\beta(\lambda_m)$ 
18       $Q \leftarrow Q \cup \{([\lambda_\ell, \lambda_m], x_\ell, x_m), ([\lambda_m, \lambda_r], x_m, x_r)\}$ 
19 return  $S$ 

```

Proof Fix some $\lambda \in [\underline{\lambda}, \bar{\lambda}]$. Then $\lambda = \gamma \underline{\lambda} + (1 - \gamma) \bar{\lambda}$ for some $\gamma \in [0, 1]$ and we can compute:

$$\begin{aligned}
 f_x(\lambda) &= a(x) + \lambda b(x) \\
 &= a(x) + (\gamma \underline{\lambda} + (1 - \gamma) \bar{\lambda}) \cdot b(x) \\
 &= \gamma \cdot (a(x) + \underline{\lambda} b(x)) + (1 - \gamma) \cdot (a(x) + \bar{\lambda} b(x)) \\
 &= \gamma \cdot f_x(\underline{\lambda}) + (1 - \gamma) \cdot f_x(\bar{\lambda}) \\
 &\leq \gamma \cdot \alpha \cdot f(\underline{\lambda}) + (1 - \gamma) \cdot \alpha \cdot f(\bar{\lambda}) \\
 &= \alpha \cdot (\gamma \cdot f(\underline{\lambda}) + (1 - \gamma) \cdot f(\bar{\lambda})) \\
 &\leq \alpha \cdot f(\lambda).
 \end{aligned}$$

Here, the first inequality follows by the assumption, and the second inequality follows from concavity of the optimal cost curve. □

The next lemma covers an important property of $\mu(\cdot, \cdot)$.

Lemma 3 *Let $\alpha \geq 1$, let $\emptyset \neq [\underline{\lambda}, \bar{\lambda}] \subseteq I$ be a compact interval and let the solutions \underline{x} and \bar{x} be α -approximate for $\underline{\lambda}$ and $\bar{\lambda}$, respectively. Assume that $b(\underline{x}) \neq b(\bar{x})$. Then the following statements are equivalent:*

- (i) *For any $\lambda \in [\underline{\lambda}, \bar{\lambda}]$, either \underline{x} or \bar{x} is α -approximate for λ .*
- (ii) *\underline{x} is α -approximate for $\mu(\underline{x}, \bar{x})$.*

(iii) \bar{x} is α -approximate for $\mu(\underline{x}, \bar{x})$.

(iv) \underline{x} is α -approximate for all $\lambda \in [\underline{\lambda}, \mu(\underline{x}, \bar{x})]$ and \bar{x} is α -approximate for all $\lambda \in [\mu(\underline{x}, \bar{x}), \bar{\lambda}]$.

Proof “(ii) \Leftrightarrow (iii)” holds since $f_{\underline{x}}(\mu(\underline{x}, \bar{x})) = f_{\bar{x}}(\mu(\underline{x}, \bar{x}))$ by Lemma 1. “(ii) \wedge (iii) \Rightarrow (iv)” follows from applying Lemma 2. “(iv) \Rightarrow (i)” is trivial. “(i) \Rightarrow (ii) \vee (iii)” follows immediately when choosing $\lambda := \mu(\underline{x}, \bar{x})$ in (i). \square

The next result justifies our choice of the initial parameter interval $[\lambda_*, \lambda^*]$ in Algorithm 1.

Proposition 1 Let $0 < \varepsilon < 1$ and $\beta \geq 1$. Define $\lambda_* \in I$ and $\lambda^* \in I$ as in Algorithm 1:

$$\lambda_* := \lambda_{\min} + \frac{1}{\beta} \cdot \varepsilon \cdot \frac{\text{LB}}{\text{UB}}, \quad (3)$$

$$\lambda^* := \lambda_{\min} + \beta \cdot \frac{1}{\varepsilon} \cdot \frac{\text{UB}}{\text{LB}}. \quad (4)$$

Then any solution $x_* \in X$ that is β -approximate for λ_* is $((1 + \varepsilon) \cdot \beta)$ -approximate for all $\lambda \leq \lambda_*$, and any solution $x^* \in X$ that is β -approximate for λ^* is $((1 + \varepsilon) \cdot \beta)$ -approximate for all $\lambda \geq \lambda^*$.

Proof We first prove the statement for λ_* . Let x_* be β -approximate for λ_* as defined in (3). Let $\lambda \in [\lambda_{\min}, \lambda_*]$ and let $x \in X$ be optimal for λ . Then $f_{x_*}(\lambda_*) \leq \beta \cdot f(\lambda_*) \leq \beta \cdot f_x(\lambda_*)$, i.e.,

$$f_{x_*}(\lambda_{\min}) + \underbrace{(\lambda_* - \lambda_{\min})}_{=\frac{1}{\beta} \cdot \varepsilon \cdot \frac{\text{LB}}{\text{UB}}} \cdot b(x_*) \leq \beta \cdot \left(f_x(\lambda_{\min}) + \underbrace{(\lambda_* - \lambda_{\min})}_{=\frac{1}{\beta} \cdot \varepsilon \cdot \frac{\text{LB}}{\text{UB}}} \cdot b(x) \right).$$

Reordering terms and using the assumption that $0 \leq b(x)$, $b(x_*) \leq \text{UB}$, we obtain

$$\begin{aligned} f_{x_*}(\lambda_{\min}) - \beta \cdot f_x(\lambda_{\min}) &\leq (\beta \cdot b(x) - b(x_*)) \cdot \frac{1}{\beta} \cdot \varepsilon \cdot \frac{\text{LB}}{\text{UB}} \\ &\leq \beta \cdot \text{UB} \cdot \frac{1}{\beta} \cdot \varepsilon \cdot \frac{\text{LB}}{\text{UB}} \\ &= \varepsilon \cdot \text{LB}. \end{aligned}$$

Now, if $f_x(\lambda_{\min}) = 0$, this means $f_{x_*}(\lambda_{\min}) \leq \varepsilon \cdot \text{LB} < \text{LB}$, so also $f_{x_*}(\lambda_{\min}) = 0$. Otherwise, if $f_x(\lambda_{\min}) \geq \text{LB}$, this implies

$$f_{x_*}(\lambda_{\min}) \leq \beta \cdot f_x(\lambda_{\min}) + \varepsilon \cdot \text{LB} \leq (\beta + \varepsilon) \cdot f_x(\lambda_{\min}) \leq (1 + \varepsilon) \cdot \beta \cdot f_x(\lambda_{\min}).$$

Since $\lambda \in [\lambda_{\min}, \lambda_*]$, the claim follows by Lemma 2.

Now, we prove the statement for λ^* . Let $\lambda > \lambda^*$ and let $x \in X$ be optimal for λ . Then $f_{x^*}(\lambda^*) \leq \beta \cdot f(\lambda^*) \leq \beta \cdot f_x(\lambda^*)$, i.e.,

$$f_{x^*}(\lambda_{\min}) + \underbrace{(\lambda^* - \lambda_{\min})}_{=\beta \cdot \frac{1}{\varepsilon} \cdot \frac{UB}{LB}} \cdot b(x^*) \leq \beta \cdot (f_x(\lambda_{\min}) + \underbrace{(\lambda^* - \lambda_{\min})}_{=\beta \cdot \frac{1}{\varepsilon} \cdot \frac{UB}{LB}} \cdot b(x)).$$

Reordering terms and using that $0 \leq f_x(\lambda_{\min}), f_{x^*}(\lambda_{\min}) \leq UB$, we obtain

$$b(x^*) - \beta \cdot b(x) \leq \frac{\beta \cdot f_x(\lambda_{\min}) - f_{x^*}(\lambda_{\min})}{\beta \cdot \frac{1}{\varepsilon} \cdot \frac{UB}{LB}} \leq \frac{\beta \cdot UB}{\beta \cdot \frac{1}{\varepsilon} \cdot \frac{UB}{LB}} = \varepsilon \cdot LB.$$

Now, if $b(x) = 0$, we get $b(x^*) \leq \varepsilon \cdot LB < LB$, so also $b(x^*) = 0$. Otherwise, if $b(x) \geq LB$, we obtain

$$b(x^*) \leq \beta \cdot b(x) + \varepsilon \cdot LB \leq (\beta + \varepsilon) \cdot b(x) \leq (1 + \varepsilon) \cdot \beta \cdot b(x).$$

Consequently, using that x^* is β -approximate for λ^* , we can conclude that

$$\begin{aligned} f_{x^*}(\lambda) &= \underbrace{f_{x^*}(\lambda^*)}_{\leq \beta \cdot f(\lambda^*)} + \underbrace{(\lambda - \lambda^*)}_{\geq 0} \cdot \underbrace{b(x^*)}_{\leq (1+\varepsilon) \cdot \beta \cdot b(x)} \\ &\leq \beta \cdot f_x(\lambda^*) + (\lambda - \lambda^*) \cdot (1 + \varepsilon) \cdot \beta \cdot b(x) \\ &\leq (1 + \varepsilon) \cdot \beta \cdot (f_x(\lambda^*) + (\lambda - \lambda^*) \cdot b(x)) \\ &= (1 + \varepsilon) \cdot \beta \cdot f_x(\lambda) \\ &= (1 + \varepsilon) \cdot \beta \cdot f(\lambda). \end{aligned}$$

□

Remark 1 In the case where $f_x(\lambda_{\min})$ and $b(x)$ only attain values from a discrete set, i.e., for all $x, x' \in X$ either $f_x(\lambda_{\min}) = f_{x'}(\lambda_{\min})$ or $|f_x(\lambda_{\min}) - f_{x'}(\lambda_{\min})| \geq LB$ and also either $b(x) = b(x')$ or $|b(x) - b(x')| \geq LB$ (e.g., in problems with $f_x(\lambda_{\min}), b(x) \in \mathbb{N}_{\geq 0}$ for all $x \in X$), we can choose λ_* and λ^* independently of ε in Proposition 1. Analogous arguments as in the proof above show that the lemma then holds true for the following values:

$$\begin{aligned} \lambda_* &:= \lambda_{\min} + \frac{1}{\beta} \cdot \frac{LB}{UB + 1} \\ \lambda^* &:= \lambda_{\min} + \beta \cdot \frac{UB + 1}{LB} \end{aligned}$$

This choice leads to a slightly better running time for Algorithm 1 as we will see in Remark 2.

The following proposition ensures that the set of solutions returned by Algorithm 1 is in fact an approximation of the desired quality. For two solutions \underline{x}, \bar{x} that are β -approximate for the two boundaries $\underline{\lambda}, \bar{\lambda}$, the proposition compares the value of the

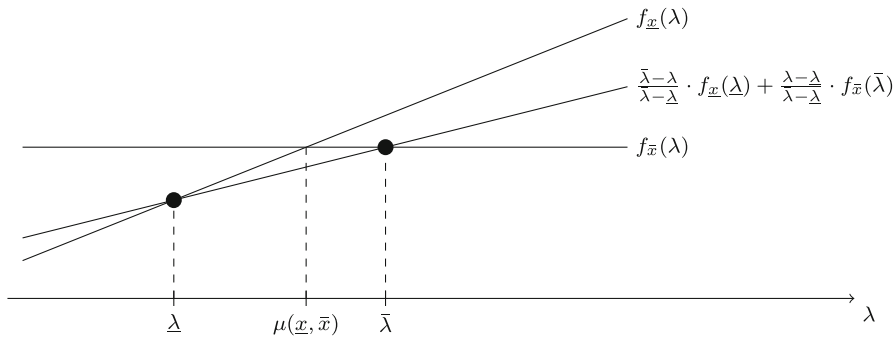


Fig. 1 Illustration of the line intersecting $f_{\underline{x}}$ at $\underline{\lambda}$ and $f_{\bar{x}}$ at $\bar{\lambda}$, which is considered in Propositions 2 and 3

unique affine-linear function that intersects $f_{\underline{x}}$ at $\underline{\lambda}$ and $f_{\bar{x}}$ at $\bar{\lambda}$ (this function does not necessarily correspond to a feasible solution $x \in X$) to the value of those two functions at their intersection point $\mu(\underline{x}, \bar{x})$ (see Fig. 1).

The proposition states that, if those two values differ by no more than a factor of α , the solutions \underline{x} and \bar{x} (together) are $(\alpha \cdot \beta)$ -approximate for the whole interval $[\underline{\lambda}, \bar{\lambda}]$. The reason for this is that, since the optimal cost curve is concave and \underline{x}, \bar{x} are β -approximate for $\underline{\lambda}$ and $\bar{\lambda}$, respectively, the connecting function is a lower bound for $\beta \cdot f(\lambda)$ within the interval $[\underline{\lambda}, \bar{\lambda}]$.

Proposition 2 Let $\alpha, \beta \geq 1$, let $\emptyset \neq [\underline{\lambda}, \bar{\lambda}] \subseteq I$ be a compact interval and let the solutions \underline{x} and \bar{x} be β -approximate for $\underline{\lambda}$ and $\bar{\lambda}$, respectively. Assume that $b(\underline{x}) \neq b(\bar{x})$, $\mu(\underline{x}, \bar{x}) \in [\underline{\lambda}, \bar{\lambda}]$, and

$$f_{\underline{x}}(\mu(\underline{x}, \bar{x})) = f_{\bar{x}}(\mu(\underline{x}, \bar{x})) \leq \alpha \cdot \left(\frac{\bar{\lambda} - \mu(\underline{x}, \bar{x})}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\underline{x}}(\underline{\lambda}) + \frac{\mu(\underline{x}, \bar{x}) - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\bar{x}}(\bar{\lambda}) \right).$$

Then \underline{x} is $(\alpha \cdot \beta)$ -approximate for all $\lambda \in [\underline{\lambda}, \mu(\underline{x}, \bar{x})]$ and \bar{x} is $(\alpha \cdot \beta)$ -approximate for all $\lambda \in [\mu(\underline{x}, \bar{x}), \bar{\lambda}]$.

Proof Lemma 3 implies that it suffices to show that \bar{x} is $(\alpha \cdot \beta)$ -approximate for $\mu(\underline{x}, \bar{x})$. First, note that we can write $\mu(\underline{x}, \bar{x})$ as

$$\begin{aligned} \frac{\bar{\lambda} - \mu(\underline{x}, \bar{x})}{\bar{\lambda} - \underline{\lambda}} \cdot \underline{\lambda} + \frac{\mu(\underline{x}, \bar{x}) - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot \bar{\lambda} &= \frac{(\underline{\lambda}\bar{\lambda} - \mu(\underline{x}, \bar{x}) \cdot \underline{\lambda}) + (\mu(\underline{x}, \bar{x}) \cdot \bar{\lambda} - \underline{\lambda}\bar{\lambda})}{\bar{\lambda} - \underline{\lambda}} \\ &= \frac{\mu(\underline{x}, \bar{x}) \cdot (\bar{\lambda} - \underline{\lambda})}{\bar{\lambda} - \underline{\lambda}} \\ &= \mu(\underline{x}, \bar{x}). \end{aligned}$$

Now, use concavity of the optimal cost curve f :

$$\begin{aligned} f_{\bar{x}}(\mu(\underline{x}, \bar{x})) &\leq \alpha \cdot \left(\frac{\bar{\lambda} - \mu(\underline{x}, \bar{x})}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\underline{x}}(\underline{\lambda}) + \frac{\mu(\underline{x}, \bar{x}) - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\bar{x}}(\bar{\lambda}) \right) \\ &\leq \alpha \cdot \left(\frac{\bar{\lambda} - \mu(\underline{x}, \bar{x})}{\bar{\lambda} - \underline{\lambda}} \cdot \beta \cdot f(\underline{\lambda}) + \frac{\mu(\underline{x}, \bar{x}) - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot \beta \cdot f(\bar{\lambda}) \right) \\ &= \alpha \cdot \beta \cdot \left(\frac{\bar{\lambda} - \mu(\underline{x}, \bar{x})}{\bar{\lambda} - \underline{\lambda}} \cdot f(\underline{\lambda}) + \frac{\mu(\underline{x}, \bar{x}) - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot f(\bar{\lambda}) \right) \\ &\leq \alpha \cdot \beta \cdot f \left(\frac{\bar{\lambda} - \mu(\underline{x}, \bar{x})}{\bar{\lambda} - \underline{\lambda}} \cdot \underline{\lambda} + \frac{\mu(\underline{x}, \bar{x}) - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot \bar{\lambda} \right) \\ &= \alpha \cdot \beta \cdot f(\mu(\underline{x}, \bar{x})). \end{aligned}$$

□

Next, we show that the condition of Proposition 2 is always fulfilled if $\mu(\underline{x}, \bar{x}) - \lambda_{\min}$ differs from either $\underline{\lambda} - \lambda_{\min}$ or $\bar{\lambda} - \lambda_{\min}$ by not more than a factor of α . This allows us to compute a polynomial bound on the number of iterations performed by Algorithm 1.

Proposition 3 *Let $\alpha \geq 1$. Let $\emptyset \neq [\underline{\lambda}, \bar{\lambda}] \subseteq I$ be a compact interval and let $\underline{x}, \bar{x} \in X$ be feasible solutions such that $f_{\underline{x}}(\underline{\lambda}) < f_{\bar{x}}(\underline{\lambda})$ and $f_{\bar{x}}(\bar{\lambda}) < f_{\underline{x}}(\bar{\lambda})$. If one of the following holds:*

- (i) $\mu(\underline{x}, \bar{x}) - \lambda_{\min} \leq \alpha \cdot (\underline{\lambda} - \lambda_{\min})$
- (ii) $\mu(\underline{x}, \bar{x}) - \lambda_{\min} \geq \frac{1}{\alpha} \cdot (\bar{\lambda} - \lambda_{\min})$

then we have

$$f_{\underline{x}}(\mu(\underline{x}, \bar{x})) \leq \alpha \cdot \left(\frac{\bar{\lambda} - \mu(\underline{x}, \bar{x})}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\underline{x}}(\underline{\lambda}) + \frac{\mu(\underline{x}, \bar{x}) - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\bar{x}}(\bar{\lambda}) \right).$$

Proof We write $\mu := \mu(\underline{x}, \bar{x})$ for short and define

$$\tilde{b} := \frac{f_{\bar{x}}(\bar{\lambda}) - f_{\underline{x}}(\underline{\lambda})}{\bar{\lambda} - \underline{\lambda}}.$$

First, note that $\mu \in [\underline{\lambda}, \bar{\lambda}]$ and $b(\underline{x}) > b(\bar{x})$. Moreover, note that

$$\begin{aligned} (\bar{\lambda} - \underline{\lambda}) \cdot \tilde{b} &= f_{\bar{x}}(\bar{\lambda}) - f_{\underline{x}}(\underline{\lambda}) \\ &= (f_{\bar{x}}(\mu) + (\bar{\lambda} - \mu) \cdot b(\bar{x})) - (f_{\underline{x}}(\mu) - (\mu - \underline{\lambda}) \cdot b(\underline{x})) \\ &= (\bar{\lambda} - \mu) \cdot b(\bar{x}) + (\mu - \underline{\lambda}) \cdot b(\underline{x}), \end{aligned}$$

which implies that $0 \leq b(\bar{x}) \leq \tilde{b} \leq b(\underline{x})$. We now prove the claim for (i). We can write

$$\frac{\bar{\lambda} - \mu}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\underline{x}}(\underline{\lambda}) + \frac{\mu - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\bar{x}}(\bar{\lambda}) = f_{\underline{x}}(\underline{\lambda}) + \tilde{b} \cdot (\mu - \underline{\lambda}),$$

so it is to show that, if (i) holds, then $f_{\underline{x}}(\mu) \leq \alpha \cdot (f_{\underline{x}}(\underline{\lambda}) + \tilde{b} \cdot (\mu - \underline{\lambda}))$. This is a simple computation:

$$\begin{aligned} f_{\underline{x}}(\mu) &= f_{\underline{x}}(\lambda_{\min}) + (\mu - \lambda_{\min}) \cdot b(\underline{x}) \\ &\leq f_{\underline{x}}(\lambda_{\min}) + \alpha \cdot (\underline{\lambda} - \lambda_{\min}) \cdot b(\underline{x}) \\ &\leq \alpha \cdot (f_{\underline{x}}(\lambda_{\min}) + (\underline{\lambda} - \lambda_{\min}) \cdot b(\underline{x})) \\ &= \alpha \cdot f_{\underline{x}}(\underline{\lambda}) \\ &\leq \alpha \cdot (f_{\underline{x}}(\underline{\lambda}) + \tilde{b} \cdot (\mu - \underline{\lambda})), \end{aligned}$$

where the first inequality follows from (i) and the second inequality holds since $f_{\underline{x}}(\lambda_{\min}) \geq 0$.

We now prove the statement for the case that (ii) holds. Similar to above, we can write

$$\frac{\bar{\lambda} - \mu}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\underline{x}}(\underline{\lambda}) + \frac{\mu - \underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \cdot f_{\bar{x}}(\bar{\lambda}) = f_{\bar{x}}(\bar{\lambda}) - \tilde{b} \cdot (\bar{\lambda} - \mu),$$

so it is to show that (ii) implies $f_{\bar{x}}(\mu) \leq \alpha \cdot (f_{\bar{x}}(\bar{\lambda}) - \tilde{b} \cdot (\bar{\lambda} - \mu))$. We first compute

$$\begin{aligned} f_{\bar{x}}(\bar{\lambda}) - (\bar{\lambda} - \lambda_{\min}) \cdot \tilde{b} &= f_{\underline{x}}(\underline{\lambda}) - (\underline{\lambda} - \lambda_{\min}) \cdot \tilde{b} \\ &\geq f_{\underline{x}}(\underline{\lambda}) - (\underline{\lambda} - \lambda_{\min}) \cdot b(\underline{x}) = f_{\underline{x}}(\lambda_{\min}) \geq 0 \end{aligned}$$

where the first equality holds since

$$(\bar{\lambda} - \lambda_{\min}) \cdot \tilde{b} - (\underline{\lambda} - \lambda_{\min}) \cdot \tilde{b} = (\bar{\lambda} - \underline{\lambda}) \cdot \tilde{b} = f_{\bar{x}}(\bar{\lambda}) - f_{\underline{x}}(\underline{\lambda}).$$

Now, we are ready to calculate

$$\begin{aligned} f_{\bar{x}}(\mu) &\leq f_{\bar{x}}(\bar{\lambda}) \\ &= \alpha \cdot \frac{1}{\alpha} \cdot f_{\bar{x}}(\bar{\lambda}) \\ &\leq \alpha \cdot \frac{\mu - \lambda_{\min}}{\bar{\lambda} - \lambda_{\min}} \cdot f_{\bar{x}}(\bar{\lambda}) \\ &= \alpha \cdot \left(f_{\bar{x}}(\bar{\lambda}) - (\bar{\lambda} - \mu) \cdot \frac{f_{\bar{x}}(\bar{\lambda})}{\bar{\lambda} - \lambda_{\min}} \right) \\ &\leq \alpha \cdot (f_{\bar{x}}(\bar{\lambda}) - (\bar{\lambda} - \mu) \cdot \tilde{b}), \end{aligned}$$

where the second inequality follows from (ii) and the last inequality follows from the previous computation. \square

We are now ready to state the main theorem of this section.

Theorem 1 Algorithm 1 returns a $((1 + \varepsilon) \cdot \beta)$ -approximation in time

$$O\left(T_{LB/UB} + T_{ALG\beta} \cdot \left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon} + \frac{1}{\varepsilon} \cdot \log \frac{UB}{LB} + \frac{1}{\varepsilon} \cdot \log \beta\right)\right),$$

where $T_{LB/UB}$ denotes the time needed for computing the bounds LB and UB, and $T_{ALG\beta}$ denotes the running time of $ALG\beta$.

Proof We first note that, at any given time during the execution of Algorithm 1, any element $([\lambda_\ell, \lambda_r], x_\ell, x_r)$ of Q satisfies $f_{x_\ell}(\lambda_\ell) \leq \beta \cdot f(\lambda_\ell)$ and $f_{x_r}(\lambda_r) \leq \beta \cdot f(\lambda_r)$.

In order to prove the approximation guarantee, let $\lambda \in I$ be chosen arbitrarily. We show the following statement: At the beginning of each iteration of the while loop starting in line 6, either $\lambda \in [\lambda_\ell, \lambda_r]$ for some element $([\lambda_\ell, \lambda_r], x_\ell, x_r)$ of Q or $f_x(\lambda) \leq (1 + \varepsilon) \cdot \beta \cdot f(\lambda)$ for some element x of S . Since $Q = \emptyset$ at termination, the approximation guarantee then follows.

At the beginning of the first iteration of the while loop, we have $Q = \{([\lambda_*, \lambda_*], x_*, x_*)\}$ and $S = \{x_*, x_*\}$. So, if $\lambda \in I \setminus [\lambda_*, \lambda_*]$, the statement follows immediately from Proposition 1. Let the statement now hold at the beginning of some iteration of the while-loop. We show that the statement holds at the end of this iteration. Let $([\lambda_\ell, \lambda_r], x_\ell, x_r)$ be the element that is extracted from Q in this iteration and assume that $\lambda \in [\lambda_\ell, \lambda_r]$ since, otherwise, there is nothing to show. We distinguish four cases: If $f_{x_\ell}(\lambda_r) \leq f_{x_r}(\lambda_r)$, then x_ℓ is added to S , which is β -approximate for $\lambda \in [\lambda_\ell, \lambda_r]$ by Lemma 2 in this case. If $f_{x_r}(\lambda_\ell) \leq f_{x_\ell}(\lambda_\ell)$, then x_r is added to S , which is then β -approximate for $\lambda \in [\lambda_\ell, \lambda_r]$ by Lemma 2. Otherwise, we know that $b(x_\ell) > b(x_r)$. If $f_{x_\ell}(\lambda_m) \leq (1 + \varepsilon) \cdot \left(\frac{\lambda_r - \lambda_m}{\lambda_r - \lambda_\ell} \cdot f_{x_\ell}(\lambda_\ell) + \frac{\lambda_m - \lambda_\ell}{\lambda_r - \lambda_\ell} \cdot f_{x_r}(\lambda_r)\right)$, then x_ℓ and x_r are added to S and the statement holds due to Proposition 2. In the remaining case, $([\lambda_\ell, \lambda_m], x_\ell, x_m)$ and $([\lambda_m, \lambda_r], x_m, x_r)$ are added to Q , and we (trivially) have $\lambda \in [\lambda_\ell, \lambda_r] = [\lambda_\ell, \lambda_m] \cup [\lambda_m, \lambda_r]$.

We prove now the bound on the running time. Starting with the initial interval $[\lambda_*, \lambda_*]$, Algorithm 1 iteratively extracts an element $([\lambda_\ell, \lambda_r], x_\ell, x_r)$ from the queue Q and checks whether the interval $[\lambda_\ell, \lambda_r]$ has to be further bisected into two subintervals $[\lambda_\ell, \lambda_m]$ and $[\lambda_m, \lambda_r]$ that need to be further explored. During this process, $ALG\beta$ is called once exactly each time an interval is bisected, computing an optimal solution x_m for the newly created interval boundary λ_m . This procedure ensures that at any given time during the execution of Algorithm 1, there does not exist a value $\lambda \in I$ for which $ALG\beta$ has been called so far and that lies in the interior of an interval that belongs to one of the current elements of Q .

First, consider the case that $I = [\lambda_{\min}, \infty)$ for some $\lambda_{\min} \in \mathbb{R}$. For all $i \in \mathbb{Z}$, define the values

$$\lambda_i := \lambda_{\min} + (1 + \varepsilon)^i. \tag{5}$$

We show that, for each $i \in \mathbb{Z}$ for which

$$\lambda_* < \lambda_{i+1} \text{ and } \lambda_i < \lambda^*, \tag{6}$$

there exists at most one value λ in the interval $[\lambda_i, \lambda_{i+1}]$ for which ALG_β is called during the execution of Algorithm 1. Note that these intervals (together) cover the initial interval $[\lambda_*, \lambda^*]$, outside of which ALG_β is never called in Algorithm 1.

Consider one of these intervals and, for the sake of a contradiction, assume that $\lambda, \lambda' \in [\lambda_i, \lambda_{i+1}]$ are two values for which ALG_β is called during the execution of Algorithm 1. We assume without loss of generality that ALG_β is not called for any value in between λ and λ' during the execution of Algorithm 1 and that ALG_β is called for λ before ALG_β is called for λ' .

Consider the iteration of the while loop during which ALG_β is called for λ' , i.e., where $\lambda' = \lambda_m = \mu(x_l, x_r)$. The value λ cannot lie in the interior of $[\lambda_\ell, \lambda_r]$ due to our previous observation. Also, λ cannot lie outside of $[\lambda_\ell, \lambda_r]$ as this would contradict our assumption that ALG_β is not called for any value in between λ and λ' during the execution of Algorithm 1 since ALG_β must have been called for λ_ℓ and λ_r earlier. Thus, we must have $\lambda_\ell = \lambda$ or $\lambda_r = \lambda$. But now, since $\lambda - \lambda_{\min}$ and $\lambda' - \lambda_{\min}$ differ by a factor of at most $1 + \varepsilon$, Proposition 3 implies that either $f_{x_\ell}(\lambda_r) \leq f_{x_r}(\lambda_r)$, $f_{x_r}(\lambda_\ell) \leq f_{x_\ell}(\lambda_\ell)$, or

$$f_{x_\ell}(\lambda') \leq (1 + \varepsilon) \cdot \left(\frac{\lambda_r - \lambda'}{\lambda_r - \lambda_\ell} \cdot f_{x_\ell}(\lambda_\ell) + \frac{\lambda' - \lambda_\ell}{\lambda_r - \lambda_\ell} \cdot f_{x_r}(\lambda_r) \right),$$

and, thus, ALG_β is not called for λ' .

It remains to count the number of values λ_i for $i \in \mathbb{Z}$ that satisfy (5) and (6), which are satisfied exactly for all $i \in \mathbb{Z}$ with

$$\log_{1+\varepsilon} \left(\frac{1}{\beta} \cdot \varepsilon \cdot \frac{\text{LB}}{\text{UB}} \right) - 1 < i < \log_{1+\varepsilon} \left(\beta \cdot \frac{1}{\varepsilon} \cdot \frac{\text{UB}}{\text{LB}} \right).$$

So, this number is in $\mathcal{O}(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon} + \frac{1}{\varepsilon} \cdot \log \frac{\text{UB}}{\text{LB}} + \frac{1}{\varepsilon} \cdot \log \beta)$. Adding the time $T_{\text{LB}/\text{UB}}$ needed for computing the bounds LB, UB at the beginning of the algorithm, this proves the bound on the running time. □

Moreover, we obtain the following corollary:

Corollary 1 *If either an exact algorithm $\text{ALG} = \text{ALG}_1$ or a (non-parametric) FPTAS is available for the non-parametric version, then Algorithm 1 yields an FPTAS. If a (non-parametric) PTAS is available for the non-parametric version, then Algorithm 1 yields a PTAS.*

Proof For the case of an exact algorithm, this follows immediately from Theorem 1. In the case of an (F)PTAS, when given an arbitrary small $\varepsilon \geq 0$, define $\delta = \sqrt{1 + \varepsilon} - 1$. Then $1 + \varepsilon = (1 + \delta) \cdot (1 + \delta)$. Theorem 1 states that we can compute a $(1 + \varepsilon)$ -approximation in time

$$\mathcal{O} \left(T_{\text{LB}/\text{UB}} + T_{\text{ALG}_{1+\delta}} \cdot \left(\frac{1}{\delta} \cdot \log \frac{1}{\delta} + \frac{1}{\delta} \cdot \log \frac{\text{UB}}{\text{LB}} \right) \right)$$

□

Remark 2 In the case of discrete problems as in Remark 1, the choice of the initial interval $[\lambda_*, \lambda^*]$ proposed in Remark 1 leads to a running time of

$$O\left(T_{LB/UB} + T_{ALG\beta} \cdot \left(\frac{1}{\varepsilon} \cdot \log \frac{UB}{LB} + \frac{1}{\varepsilon} \cdot \log \beta\right)\right).$$

4 Small cardinality

In this section, we address the issue of computing approximations for parametric optimization problems that are as small as possible in terms of cardinality. To this end, we provide an algorithm that, given an α -approximation R and an exact algorithm $ALG = ALG_1$ for the non-parametric problem, returns an α -approximation S that has minimum cardinality among the ones contained in R . If it is known that R consists of solutions that are supported in R , we show how this information can be used in order to reduce the running time of this algorithm significantly.

Moreover, if $S^* \subseteq X$ is an α -approximation of minimum cardinality for a given instance of a parametric optimization problem, we show that this algorithm can be used along with Algorithm 1 of the previous section in order to compute an α -approximation S of cardinality $|S| \leq 2 \cdot |S^*|$.

Throughout this section, we assume that ALG can not only find an optimal solution for any $\lambda \in I$ but can also compute a lexicographically optimal solution for the (non-parametric) optimization problem

$$\text{lexmin}_{x \in X}(b(x), a(x)), \tag{7}$$

where we assume that this optimum exists. For $\alpha \geq 1$, we say that a solution $x \in X$ is α -approximate for (7) if, for an optimal solution x^* of (7), we have $b(x) \leq \alpha \cdot b(x^*)$ and $a(x) \leq \alpha \cdot a(x^*)$.

Solving (7) corresponds to minimizing $a + \lambda \cdot b$ for “ $\lambda \rightarrow \infty$ ” in the following sense: A solution $x \in X$ is optimal for (7) if and only if, for any $x' \in X$, there exists some $\lambda^* \in I$ such that $f_x(\lambda) \leq f_{x'}(\lambda)$ for all $\lambda \geq \lambda^*$. This also applies to approximation: A solution $x \in X$ is α -approximate for (7) if and only if, for any $x' \in X$, there exists some $\lambda^* \in I$ such that $f_x(\lambda) \leq \alpha \cdot f_{x'}(\lambda)$ for all $\lambda \geq \lambda^*$.

For convenience, we define the following notation: We write $f_x(\infty) := (b(x), a(x))$ for all $x \in X$ and $f(\infty) := f_{x^*}(\infty)$ for an optimal solution $x^* \in X$ to (7). When writing $f_x(\infty) \leq f_{x'}(\infty)$ for two solutions $x, x' \in X$, we refer to the lexicographical order relation. We also write “ $ALG(\infty)$ ” when indicating that ALG is called for solving (7).

The additional assumption that ALG is able to solve (7) is a reasonable one since, for most applications of linear parametric optimization, the functions a and b admit values in a discrete set such that the values of any two different solutions $x, x' \in X$ are either equal or differ by at least LB . In this case, (7) is equivalent to the non-parametric problem for $\lambda > \frac{UB}{LB}$. Thus, solving (7) is typically not more difficult than minimizing the objective function $a + \lambda \cdot b$ for any other $\lambda \in I$.

We stress the fact that this assumption is necessary as the following theorem states.

Theorem 2 For a given $\alpha > 1$, no deterministic algorithm can compute a minimum cardinality subset R^* of an α -approximation R that maintains α as the approximation guarantee for the general linear parametric problem (1) when given as the input only:

- the α -approximation R
- the interval $I = [\lambda_{\min}, \infty)$
- a routine ALG that computes an optimal solution for any $\lambda \in I$ but not for (7)

Proof For the sake of a contradiction, assume that there exists such a deterministic algorithm A that, for any instance of a linear parametric optimization problem, terminates after finitely many calls to ALG. Consider the following instance: $I = [0, \infty)$ and $X = \{x_1, x_2, x_3\}$ with

$$\begin{aligned} f_{x_1}(\lambda) &= \alpha + \lambda \cdot \alpha \\ f_{x_2}(\lambda) &= (\alpha + 1) + \lambda \\ f_{x_3}(\lambda) &= 1 + \lambda \end{aligned}$$

When A is used to compute a minimum cardinality α -approximation that is a subset of the α -approximation $R = \{x_1, x_2\}$ it returns $\{x_1\}$ after finitely many calls to ALG. Since, for any $\lambda \in I$, x_3 is the only optimal solution, all of these calls to ALG return x_3 . We denote by $\bar{\lambda}$ the maximum value for which ALG is called by A in this scenario. We assume, without loss of generality, that $\bar{\lambda} > 0$. Otherwise set $\bar{\lambda} = 1$.

Now consider the instance with $I = [0, \infty)$ and $X = \{x_1, x_2, x_3, x_4\}$, where

$$f_{x_4}(\lambda) = (\alpha + 1) + \lambda \cdot \max \left\{ 1 - \frac{\alpha}{\bar{\lambda} + \alpha}, \frac{1}{\alpha} \right\},$$

(see also Fig. 2) and again, use A to compute a minimum cardinality subset of $R = \{x_1, x_2\}$. Note that R is, in fact, an α -approximation in this instance since, for all $\lambda \in I$,

$$f_{x_2}(\lambda) = (\alpha + 1) + \lambda < \alpha \cdot \left((\alpha + 1) + \lambda \cdot \frac{1}{\alpha} \right) \leq \alpha \cdot f_{x_4}(\lambda).$$

Also, note that the input for A is exactly the same in this scenario as in the previous one. Moreover, for all $0 < \lambda \leq \bar{\lambda}$, we have

$$\begin{aligned} f_{x_3}(\lambda) &= 1 + \lambda \\ &\leq 1 + \lambda + \alpha - \frac{\lambda \cdot \alpha}{\bar{\lambda}} \\ &< 1 + \lambda + \alpha - \frac{\lambda \cdot \alpha}{\bar{\lambda} + \alpha} \\ &= (\alpha + 1) + \lambda \cdot \left(1 - \frac{\alpha}{\bar{\lambda} + \alpha} \right) \\ &\leq f_{x_4}(\lambda), \end{aligned}$$

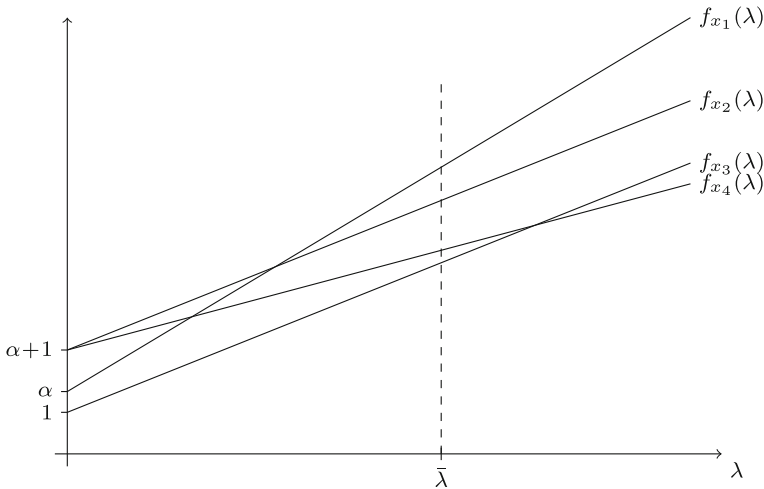


Fig. 2 Illustration of the instance constructed in the proof of Theorem 2. The solution x_3 is optimal for all $\lambda \leq \bar{\lambda}$. The solution x_1 is not α -approximate for $\lambda > \mu(x_3, x_4)$

and we also have $f_{x_3}(0) = 1 < 1 + \alpha = f_{x_4}(0)$, so ALG returns x_3 , i.e., the same solution as in the other instance, for all $\lambda \leq \bar{\lambda}$. Thus, since A is a deterministic algorithm, it behaves in exactly the same way in both of these scenarios. It returns $\{x_1\}$ after doing the same calls to ALG as before.

However, this is not a correct output since $\{x_1\}$ is not an α -approximation: For $\lambda > \max\{\bar{\lambda} + \alpha, \frac{\alpha^2}{\alpha-1}\}$, we have

$$\begin{aligned} f_{x_1}(\lambda) &= \alpha \cdot (1 + \lambda) \\ &> \alpha \cdot \left(1 + \lambda + \alpha \cdot \left(1 - \frac{\lambda}{\bar{\lambda} + \alpha}\right)\right) \\ &= \alpha \cdot \left((\alpha + 1) + \lambda \cdot \left(1 - \frac{\alpha}{\bar{\lambda} + \alpha}\right)\right) \end{aligned}$$

and

$$\begin{aligned} f_{x_1}(\lambda) &= \alpha \cdot (1 + \lambda) \\ &= \alpha \cdot \left(1 + \frac{\lambda}{\alpha} + \frac{(\alpha - 1) \cdot \lambda}{\alpha}\right) \\ &> \alpha \cdot \left(1 + \frac{\lambda}{\alpha} + \frac{\alpha^2}{\alpha}\right) \\ &= \alpha \cdot \left((\alpha + 1) + \lambda \cdot \frac{1}{\alpha}\right) \end{aligned}$$

and, thus, $f_{x_1}(\lambda) > \alpha \cdot f_{x_4}(\lambda) = \alpha \cdot f(\lambda)$. □

Algorithm 2 describes the following greedy procedure that, given a β -approximation $R \subseteq X$, extracts a minimum cardinality β -approximation $S \subseteq R$ from R : We start by choosing the solution among R that covers the largest set of parameter values containing λ_{\min} in terms of β -approximation, i.e., that is β -approximate for the maximum value $\lambda \in I$ while still being β -approximate for λ_{\min} . After that, we always choose the solution that, together with all the previously chosen solutions, covers the largest connected portion of I (starting at λ_{\min}) in terms of β -approximation.

The algorithm takes advantage of the following property of μ : If two solutions $x, x' \in X$ are β -approximate at $\mu(x, x')$, then the one of them having the smaller slope covers a larger portion of $[\mu(x, x'), \infty)$. If x, x' are not β -approximate at $\mu(x, x')$, then the one of them having the larger slope covers a larger portion of $[\lambda_{\min}, \mu(x, x'))$ (if one of them covers any value in this interval at all).

This property suffices to be able to compare two solutions $x, x' \in R$ with respect to the above criterion in all necessary cases using a single ALG-call.

The following theorem formally states the correctness and running time of Algorithm 2.

Algorithm 2: Algorithm minimizing the cardinality of an α -approximation

input : An α -approximation $R = \{x_1, \dots, x_k\}$, an exact algorithm ALG for the non-parametric version of Π

output: A smallest-cardinality subset of R that is an α -approximation

- 1 Sort x_1, \dots, x_k such that $f_{x_1}(\lambda_{\min}) \leq f_{x_2}(\lambda_{\min}) \leq \dots \leq f_{x_k}(\lambda_{\min})$ and $f_{x_i}(\infty) \leq f_{x_{i+1}}(\infty)$ if $f_{x_i}(\lambda_{\min}) = f_{x_{i+1}}(\lambda_{\min})$
- 2 If $f_{x_i}(\infty) \leq f_{x_{i+1}}(\infty)$ for some i , delete x_{i+1} until $f_{x_1}(\lambda_{\min}) < f_{x_2}(\lambda_{\min}) < \dots < f_{x_k}(\lambda_{\min})$ and $f_{x_1}(\infty) > f_{x_2}(\infty) > \dots > f_{x_k}(\infty)$
- 3 $x_{\min} \leftarrow \text{ALG}(\lambda_{\min}); x_{\infty} \leftarrow \text{ALG}(\infty)$
- 4 $S \leftarrow \emptyset;$ /* solution set */
- 5 $i \leftarrow 1;$ /* index of currently found best next solution */
- 6 $j \leftarrow 2;$ /* index of current solution */
- 7 $\pi \leftarrow 0;$ /* index of previously chosen solution */
- 8 **while** $f_{x_j}(\lambda_{\min}) \leq \alpha \cdot f_{x_{\min}}(\lambda_{\min})$ and $j \leq k$ **do**
- 9 **if** $f_{x_j}(\mu(x_i, x_j)) \leq \alpha \cdot f(\mu(x_i, x_j))$ **then**
- 10 $i \leftarrow j$
- 11 $j \leftarrow j + 1$
- 12 $S \leftarrow \{x_i\}$
- 13 **while** $f_{x_i}(\infty) > \alpha \cdot f_{x_{\infty}}(\infty)$ **do**
- 14 $\pi \leftarrow i$
- 15 **foreach** $j = \pi + 1, \dots, k$ **do**
- 16 **if** $f_{x_j}(\mu(x_{\pi}, x_j)) \leq \alpha f(\mu(x_{\pi}, x_j))$ and $f_{x_j}(\mu(x_i, x_j)) \leq \alpha f(\mu(x_i, x_j))$ **then**
- 17 $i \leftarrow j$
- 18 $S \leftarrow S \cup \{x_i\}$
- 19 **return** S

Theorem 3 Algorithm 2 computes a minimum cardinality subset of R that is an α -approximation using $\mathcal{O}(k^2)$ calls to ALG.

Proof In this proof, we use the following notation: For $\iota = 1, \dots, k$, we define $\bar{\lambda}_{x_\iota}$ to be the maximum parameter value for which the solution $x_\iota \in R$ (after the sorting and the deleting step) is α -approximate, i.e.,

$$\bar{\lambda}_{x_\iota} := \max\{\lambda \in I : f_{x_\iota}(\lambda) \leq \alpha \cdot f(\lambda)\}. \tag{8}$$

We use the convention that $\bar{\lambda}_{x_\iota} = -\infty$ if the set in this definition is empty or if $\iota = 0$, and that $\bar{\lambda}_{x_\iota} = \infty$ if the set in this definition is unbounded. If $\bar{\lambda}_{x_\iota} \in I$ for some $x_\iota \in X$, then $f_{x_\iota}(\bar{\lambda}_{x_\iota}) = \alpha \cdot f(\bar{\lambda}_{x_\iota})$ since f and f_{x_ι} are continuous.

Moreover, for any two solutions $x_i, x_j \in R$ with $b(x_i) > b(x_j)$, if there exists some $\lambda \leq \mu(x_i, x_j)$ for which both x_i and x_j are α -approximate, we have $\bar{\lambda}_{x_j} \geq \bar{\lambda}_{x_i}$ if and only if x_j is α -approximate for $\mu(x_i, x_j)$ (and, thus, also x_i is α -approximate for $\mu(x_i, x_j)$): If x_i and x_j are α -approximate for $\mu(x_i, x_j)$, then $\bar{\lambda}_{x_i}, \bar{\lambda}_{x_j} \geq \mu(x_i, x_j)$. Thus, by Lemma 1,

$$f_{x_j}(\bar{\lambda}_{x_i}) \leq f_{x_i}(\bar{\lambda}_{x_i}) \leq \alpha \cdot f(\bar{\lambda}_{x_i}),$$

which implies that $\bar{\lambda}_{x_j} \geq \bar{\lambda}_{x_i}$. Similarly, if x_i and x_j are not α -approximate for $\mu(x_i, x_j)$, then $\bar{\lambda}_{x_i}, \bar{\lambda}_{x_j} < \mu(x_i, x_j)$. In this case, we have $\bar{\lambda}_{x_j} < \bar{\lambda}_{x_i}$ since, by Lemma 1,

$$f_{x_j}(\bar{\lambda}_{x_i}) > f_{x_i}(\bar{\lambda}_{x_i}) = \alpha \cdot f(\bar{\lambda}_{x_i}).$$

We now prove that Algorithm 2 terminates after performing $\mathcal{O}(k^2)$ calls to ALG. Apart from the initialization, ALG is only called when evaluating the optimal cost curve f at $\mu(x_i, x_j)$ or $\mu(x_\pi, x_j)$ in lines 9 and 16. In the while loop starting in line 8, at most k iterations are performed.

The stopping condition of the while loop starting in line 13 is also fulfilled after at most k iterations: We know that R is an α -approximation, so, for any $x_i \in R$, we either have $f_{x_i}(\infty) \leq \alpha \cdot f(\infty)$ or there exists an $x_j \in R$ with $j > i$ such that $f_{x_j}(\mu(x_i, x_j)) \leq \alpha \cdot f(\mu(x_i, x_j))$. Note that in the beginning of each iteration of the while loop, π is set to be equal to i , so the condition in line 16 simplifies to $f_{x_j}(\mu(x_i, x_j)) \leq \alpha \cdot f(\mu(x_i, x_j))$. Thus, the value of i is strictly increased in each iteration. When $i = k$ (at the latest), the algorithm terminates since R is an α -approximation, so we must have $f_{x_i}(\infty) \leq \alpha \cdot f(\infty)$ for some $x_i \in R$.

In each iteration of the second while loop, $2 \cdot (k - \pi) \in \mathcal{O}(k)$ many calls to ALG are performed.

We now show that the set S returned by Algorithm 2 is indeed an α -approximation. To this end, we show by induction that at any time during the execution of the algorithm,

- (i) S is an α -approximation for all $\lambda \in [\lambda_{\min}, \bar{\lambda}_{x_\pi}]$ and
- (ii) the set $S \cup \{x_i\}$ is an α -approximation for all $[\lambda_{\min}, \bar{\lambda}_{x_i}]$

Statements (i) and (ii) obviously hold after the initialization, when $S = \emptyset, i = 1$ and $\pi = 0$. Afterwards, whenever x_i is added to S , π is set to be equal to i , so it suffices to show (ii), since this immediately implies (i) at any given time during the execution

of the algorithm. We assume now that statements (i) and (ii) hold immediately before the value of i is changed during the execution of the algorithm and show that (ii) holds after this step.

When i is assigned a new value in line 10, we know that (for this new value)

$$f_{x_i}(\lambda_{\min}) \leq \alpha \cdot f_{x_{\min}}(\lambda_{\min}) = \alpha \cdot f(\lambda_{\min})$$

by line 8, which, in particular, implies $\bar{\lambda}_{x_i} \geq \lambda_{\min}$. Since also

$$f_{x_i}(\bar{\lambda}_{x_i}) \leq \alpha \cdot f(\bar{\lambda}_{x_i})$$

by (8), statement (ii) follows from Lemma 2.

When i is assigned a new value in line 17 and if statement (i) holds at this moment, it suffices to show that x_i is α -approximate for all $\lambda \in [\bar{\lambda}_{x_\pi}, \bar{\lambda}_{x_i}]$. We know that

$$f_{x_\pi}(\mu(x_\pi, x_i)) = f_{x_i}(\mu(x_\pi, x_i)) \leq \alpha \cdot f(\mu(x_\pi, x_i)),$$

so $\bar{\lambda}_{x_\pi} \geq \mu(x_\pi, x_i)$ and also $\bar{\lambda}_{x_i} \geq \mu(x_\pi, x_i)$. Since we also have $f_{x_i}(\bar{\lambda}_{x_i}) \leq \alpha \cdot f(\bar{\lambda}_{x_i})$ by (8), Lemma 2 implies that x_i is α -approximate for all $\lambda \in [\bar{\lambda}_{x_\pi}, \bar{\lambda}_{x_i}] \subseteq [\mu(x_\pi, x_i), \bar{\lambda}_{x_i}]$.

When the algorithm terminates, we know that $f_{x_i}(\infty) \leq \alpha \cdot f_{x_\infty}(\infty)$, so we have $\bar{\lambda}_{x_i} = \infty$. Since x_i must have been added to S immediately before, this concludes the proof that the returned set S is an α -approximation.

Finally, we show that S has minimum cardinality among all α -approximations $S' \subseteq R$. Note that we only have to consider the remaining set after the sorting and deleting in the beginning of the algorithm, i.e., it suffices to show that S has minimum cardinality among all α -approximations $S' \subseteq \{x_1, \dots, x_k\}$, since if, for two solutions $x, x' \in R$, we have $f_x(\lambda_{\min}) \leq f_{x'}(\lambda_{\min})$ and $f_x(\infty) \leq f_{x'}(\infty)$, x' can be replaced by x in any α -approximation. Thus, in the following, we assume that $R = \{x_1, \dots, x_k\}$. We know that $S = \{x_{j_1}, \dots, x_{j_\ell}\}$ for some $j_1 < \dots < j_\ell$ and $l = |S|$, and we denote the cardinality of a smallest α -approximation $S^* \subseteq R$ by $m := |S^*|$.

It is easy to see that, for the ordering of x_1, \dots, x_k , it holds that $b(x_1) > \dots > b(x_k)$. Moreover, we note that the fact that the value of i is never decreased during the execution of the algorithm implies that $x_{j_1}, \dots, x_{j_\ell}$ are added to S in exactly this order.

We show the following statement via induction on $\iota = 1, \dots, \ell$: There exists a subset S_ι^* of R such that

- S_ι^* is an α -approximation,
- S_ι^* has cardinality m , and
- S_ι^* contains $\{x_{j_1}, \dots, x_{j_\iota}\}$.

For $\iota = \ell$, this statement implies, that $m \leq |S| \leq |S_\ell^*| = m$, i.e., S also has cardinality m .

In order to show the statement for $\iota = 1$, let $S_0^* = \{x_{j_1^*}, \dots, x_{j_m^*}\} \subseteq R$ with $j_1^* < \dots < j_m^*$ be an α -approximation. We show that x_{j_1} is α -approximate for any $\lambda \in [\lambda_{\min}, \mu(x_{j_1^*}, x_{j_2^*})]$ and we can therefore replace $x_{j_1^*}$ by x_{j_1} in S_0^* : Note that

whenever the condition in line 9 is checked during the execution of the algorithm, we know that $i < j$, i.e., $b(x_i) > b(x_j)$. Moreover, both x_i and x_j are α -approximate for λ_{\min} , which implies that the condition that $f_{x_j}(\mu(x_i, x_j)) \leq \alpha \cdot f(\mu(x_i, x_j))$, which is checked in line 9, is equivalent to the condition that $\bar{\lambda}_{x_j} \geq \bar{\lambda}_{x_i}$. Thus, after the while loop starting in line 8, x_i satisfies $\bar{\lambda}_{x_i} \geq \bar{\lambda}_{x_j}$ for all $x_j \in R$ with $f_{x_j}(\lambda_{\min}) \leq \alpha \cdot f(\lambda_{\min})$, so, in particular, we have

$$\bar{\lambda}_{x_{j_1}} = \bar{\lambda}_{x_i} \geq \bar{\lambda}_{x_{j_1}^*} \geq \mu(x_{j_1}^*, x_{j_2}^*),$$

where the equality holds since $x_i = x_{j_1}$ is the first solution added to S and the last inequality holds since $x_{j_1}^*$ is α -approximate for $\mu(x_{j_1}^*, x_{j_2}^*)$ in S_0^* .

Next, we do the induction step $\iota \rightarrow \iota + 1$ for $\iota = 1, \dots, \ell - 1$. To this end, let the set $S_\iota^* = \{x_{j_1}^*, \dots, x_{j_m}^*\} \subseteq R$ with $j_1^* < \dots < j_m^*$ be an α -approximation of cardinality m that contains $\{x_{j_1}, \dots, x_{j_\iota}\}$. Note that we must have $x_{j_1}^* = x_{j_1}, \dots, x_{j_\iota}^* = x_{j_\iota}$ since S is an α -approximation and S_ι^* has minimum cardinality.

Consider the iteration of the while loop starting in line 13 in which $x_{j_{\iota+1}}$ is added to S . In this iteration, we have $\pi = j_\iota = j_\iota^*$. Note that the condition checked in line 16 is equivalent to

$$f_{x_j}(\mu(x_\pi, x_j)) \leq \alpha \cdot f(\mu(x_\pi, x_j)) \text{ and } \bar{\lambda}_{x_j} \geq \bar{\lambda}_{x_i},$$

so, at the end of the inner loop starting in line 15, x_i satisfies $\bar{\lambda}_{x_i} \geq \bar{\lambda}_{x_j}$ for all $j \geq \pi$ with $f_{x_j}(\mu(x_\pi, x_j)) \leq \alpha \cdot f(\mu(x_\pi, x_j))$. Moreover, x_i is added to S at the end of this iteration, so we have $i = j_{\iota+1}$. Thus, we can conclude $\bar{\lambda}_{x_{j_{\iota+1}}} = \bar{\lambda}_{x_i} \geq \bar{\lambda}_{x_{j_{\iota+1}}^*}$. Again, we argue that we can replace $x_{j_{\iota+1}}^*$ by $x_{j_{\iota+1}}$ in S_ι^* :

If $\iota = 1$, since S is an α -approximation, $x_{j_\iota} = x_{j_\iota}^* \in S_\iota^*$ is α -approximate for any

$$\lambda \in [\lambda_{\min}, \mu(x_{j_\iota}, x_{j_{\iota+1}})] = [\lambda_{\min}, \mu(x_{j_\iota}^*, x_{j_{\iota+1}})].$$

Similarly, if $\iota \in \{2, \dots, \ell - 1\}$, x_{j_ι} is α -approximate for any

$$\lambda \in [\mu(x_{j_{\iota-1}}, x_{j_\iota}), \mu(x_{j_\iota}, x_{j_{\iota+1}})] = [\mu(x_{j_{\iota-1}}^*, x_{j_\iota}^*), \mu(x_{j_\iota}^*, x_{j_{\iota+1}})].$$

Moreover, $x_{j_{\iota+1}}$ is α -approximate for any

$$\lambda \in [\mu(x_{j_\iota}, x_{j_{\iota+1}}), \bar{\lambda}_{x_{j_{\iota+1}}}] \supseteq [\mu(x_{j_\iota}^*, x_{j_{\iota+1}}), \bar{\lambda}_{x_{j_{\iota+1}}^*}].$$

This concludes the proof that $|S| = m$. □

Remark 3 For $\beta > \alpha \geq 1$, any α -approximation is also a β -approximation. Thus, letting Algorithm 2 interpret a given α -approximation S_α as a β -approximation yields a β -approximation $S_\beta \subseteq S_\alpha$ that has minimum cardinality among all subsets of S_α that are a β -approximation.

In Algorithm 2, if we know that all solutions $x_1, \dots, x_k \in R$ are supported, we can reduce the running time by modifying the second while loop using the following property:

Lemma 4 *Let $\alpha \geq 1$, let $R \subseteq X$ be an α -approximation, and let $x, x', x'' \in X$ be solutions that are supported in R such that $b(x) > b(x') > b(x'')$. If*

$$f_{x'}(\mu(x, x')) > \alpha \cdot f(\mu(x, x'))$$

then also

$$f_{x''}(\mu(x, x'')) > \alpha \cdot f(\mu(x, x'')).$$

Proof Since x is supported in R , there exists $\lambda_x \in R$ such that $f_x(\lambda_x)$ is minimum within R , i.e., where, in particular, $f_x(\lambda_x) \leq f_{x'}(\lambda_x)$ and $f_x(\lambda_x) \leq \alpha \cdot f(\lambda_x)$. Therefore, by Lemma 2, it suffices to show that $\lambda_x \leq \mu(x, x') \leq \mu(x, x'')$. The first inequality follows immediately from Lemma 1. In order to prove the second inequality, assume, for the sake of a contradiction, that $\mu(x, x') > \mu(x, x'')$. Then, by Lemma 1,

$$f_{x'}(\mu(x, x'')) > f_x(\mu(x, x'')) = f_{x''}(\mu(x, x''))$$

and, therefore, for all $\lambda \leq \mu(x, x'')$,

$$\begin{aligned} f_{x'}(\lambda) &= f_{x'}(\mu(x, x'')) + (\lambda - \mu(x, x'')) \cdot b(x') \\ &> f_{x''}(\mu(x, x'')) + (\lambda - \mu(x, x'')) \cdot b(x'') \\ &= f_{x''}(\lambda), \end{aligned}$$

and, for $\lambda \geq \mu(x, x'')$,

$$\begin{aligned} f_{x'}(\lambda) &= f_{x'}(\mu(x, x'')) + (\lambda - \mu(x, x'')) \cdot b(x') \\ &> f_x(\mu(x, x'')) + (\lambda - \mu(x, x'')) \cdot b(x) \\ &= f_x(\lambda), \end{aligned}$$

which contradicts the fact that x' is supported in R . \square

Due to this property, we know that, during the execution of Algorithm 2, whenever $f_{x_j}(\mu(x_\pi, x_j)) > \alpha \cdot f(\mu(x_\pi, x_j))$ for some j in line 16, this also holds for $j + 1, \dots, k$. Therefore, we do not have to check the first condition of line 16 for all $j > \pi$ in each iteration of the while loop starting in line 13. The second condition can be removed completely in the case that all solutions in R are supported in R , since in this case, for $j = 1, \dots, k - 1$, we have that x_j is α -approximate for $(\mu(x_j, x_j + 1))$.

In Algorithm 2, if all solutions $x \in R$ are supported in R , we can, thus, replace lines 15 through 17 by the following without changing the output of the algorithm:

Moreover, for the same reason that the second condition of line 16 in the original version of Algorithm 2 is always fulfilled, the condition in line 9 is also always fulfilled and can therefore be removed. We obtain the following result:

```

15 while  $f_{x_j}(\mu(x_\pi, x_j)) \leq \alpha \cdot f(\mu(x_\pi, x_j))$  do
16    $i \leftarrow j$ 
17    $j \leftarrow j + 1$ 
    
```

Theorem 4 *If R is an α -approximation consisting of solutions that are supported in R , the modified version of Algorithm 2 computes a minimum cardinality subset of R that is an α -approximation using $\mathcal{O}(k)$ calls to ALG.*

Proof As argued above, the output of Algorithm 2 is not changed by the modifications if R consists of solutions that are supported in R .

Moreover, since, in the modified algorithm, f is evaluated only for values $\mu(x_\pi, x_j)$, where, after each evaluation, either π or j is strictly increased, the number of evaluations is in $\mathcal{O}(k)$. □

Given an instance of a parametric optimization problem and α , one might be interested in computing α -approximations containing the minimum number of solutions needed for an α -approximation. When having access to an algorithm ALG solving the non-parametric problem exactly, we are able to compute approximations consisting of solutions that are supported not only in the approximation itself but in X . The following results state that any α -approximation consisting of supported solutions contains an α -approximation whose cardinality is at most twice as large as the minimum cardinality of any α -approximation for the given instance.

Lemma 5 *Let $\alpha \geq 1$ and let $S = \{x_1, \dots, x_k\}$ be an α -approximation consisting of supported solutions $x_i \in X$, where $b(x_1) > b(x_2) > \dots > b(x_k)$. Let $[\underline{\lambda}, \bar{\lambda}] \subseteq I$ be an interval and let $x^* \in X$ be some solution that is α -approximate for all $\lambda \in [\underline{\lambda}, \bar{\lambda}]$. Then there exist two (or fewer) solutions $x, x' \in S$ such that, for each $\lambda \in [\underline{\lambda}, \bar{\lambda}]$, either x is α -approximate or x' is α -approximate.*

Proof Since S consists of supported solutions, we know that, for each $x_i \in S$, there exists a value $\lambda_i \in I$ such that $f_{x_i}(\lambda_i) = f(\lambda_i)$. For these values, Lemma 1 implies that

$$\lambda_1 \leq \mu(x_1, x_2) \leq \lambda_2 \leq \dots \leq \lambda_{k-1} \leq \mu(x_{k-1}, x_k) \leq \lambda_k.$$

First, consider the case that $b(x^*) > b(x_1)$. By Lemma 1, we know that $f_{x^*}(\lambda) < f_{x_1}(\lambda)$ for all $\lambda < \mu(x^*, x_1)$, so we must have $\lambda_1 \geq \mu(x^*, x_1)$.

We know that x_1 is α -approximate for all $\lambda \in I$ with $\lambda \leq \mu(x_1, x_2)$, so, in particular, x_1 is α -approximate for all $\lambda \in [\underline{\lambda}, \mu(x_1, x^*)]$. For $\lambda \in [\mu(x_1, x^*), \bar{\lambda}]$, we have $f_{x_1}(\lambda) \leq f_{x^*}(\lambda)$ by Lemma 1, so, since x^* is α -approximate for λ , also x_1 is α -approximate for λ . Similarly, we can show that, in the case that $b(x^*) < b(x_k)$, x_k is α -approximate for all $\lambda \in [\underline{\lambda}, \bar{\lambda}]$.

Otherwise, we can choose i maximal such that $b(x_i) \geq b(x^*)$ and $b(x_{i+1}) < b(x^*)$. Then x_i is α -approximate for all $\lambda \in [\lambda_i, \mu(x_i, x_{i+1})]$ and x_{i+1} is α -approximate

for all $\lambda \in [\mu(x_i, x_{i+1}), \lambda_{i+1}]$. Moreover, if $\underline{\lambda} < \lambda_i$, then x_i is α -approximate for all $\lambda \in [\underline{\lambda}, \lambda_i]$: Let $\lambda \in [\underline{\lambda}, \lambda_i]$. Then

$$\begin{aligned} f_{x_i}(\lambda) &= \underbrace{f_{x_i}(\lambda_i)}_{=f(\lambda_i)} + \underbrace{(\lambda - \lambda_i)}_{\leq 0} \cdot \underbrace{b(x_i)}_{\geq b(x^*)} \\ &\leq f_{x^*}(\lambda_i) + (\lambda - \lambda_i) \cdot b(x^*) \\ &= f_{x^*}(\lambda) \\ &\leq \alpha \cdot f(\lambda). \end{aligned}$$

Analogously, if $\bar{\lambda} > \lambda_{i+1}$, then x_{i+1} is α -approximate for all $\lambda \in [\lambda_{i+1}, \bar{\lambda}]$. \square

Note that Lemma 5 also holds for intervals $[\underline{\lambda}, \infty) \subseteq I$. Thus, we have the following corollary:

Corollary 2 *Let $S \subseteq X$ be an α -approximation consisting of supported solutions and let $S^* \subseteq X$ be an α -approximation having minimum cardinality among all α -approximations for the given instance. Then there exists an α -approximation $S' \subseteq S$ with $|S'| \leq 2 \cdot |S^*|$.*

Proof By Lemma 5, for each solution $x^* \in S^*$, there exist two or fewer solutions in S such that, for all $\lambda \in I$ for which x^* is α -approximate, at least one of the two is α -approximate. Thus, for an α -approximation, at most $2 \cdot |S^*|$ solutions from S are necessary. \square

Since, when using an exact algorithm ALG for the non-parametric problem, Algorithm 1 computes a $(1 + \varepsilon)$ -approximation consisting of supported solutions, we obtain the following:

Corollary 3 *For $\varepsilon > 0$, let S^* be a $(1 + \varepsilon)$ -approximation of minimum cardinality. We can compute a $(1 + \varepsilon)$ -approximation S of cardinality $|S| \leq 2 \cdot |S^*|$ using $\mathcal{O}(\frac{1}{\varepsilon} \cdot \log(\frac{1}{\varepsilon}) + \frac{1}{\varepsilon} \cdot \log(\text{UB}))$ evaluations of ALG.*

Proof Apply Algorithm 1 and the modified version of Algorithm 2 consecutively. The claim follows from Theorem 1, Corollary 2, and Theorem 4. \square

Remark 4 In Algorithm 1, when using a stack instead of a queue in order to handle the remaining intervals, the solutions x of the returned set are computed in decreasing order of $b(x)$. This means that, when applying Algorithm 1 and Algorithm 2 consecutively, the sorting step of Algorithm 2 can be skipped if this ordering is preserved.

The following example shows that there exist instances of linear parametric optimization problems where the unique minimum cardinality α -approximation does not contain any supported solutions. This means that no algorithm that can only access the instance via an ALG routine is able to compute a minimum cardinality α -approximation and the factor of two achieved by Corollary 3 is, in this sense, best possible. Moreover, the example can be used to demonstrate that, in the modified version of Algorithm 2, the condition that the input set R consists of solutions that are supported in R is, in fact, necessary.

Example 1 For $\alpha > 1$, consider an instance with $X = \{x_1, x_2, x_3\}$, $I = [0, \infty)$, and

$$\begin{aligned} a(x_1) &= 2 & b(x_1) &= 3\alpha - 1 \\ a(x_2) &= 2\alpha & b(x_2) &= 2\alpha \\ a(x_3) &= 3\alpha - 1 & b(x_3) &= 2 \end{aligned}$$

Then, for $\lambda \leq 1$, we have $f_{x_2}(\lambda) > f_{x_1}(\lambda)$ and for $\lambda \geq 1$, we have $f_{x_2}(\lambda) > f_{x_3}(\lambda)$, so x_2 is not supported. Moreover, we have

$$f_{x_3}(0) = 3\alpha - 1 > 2\alpha = \alpha \cdot f_{x_1}(0),$$

and

$$\begin{aligned} f_{x_1}(3\alpha + 3) &= 2 + (3\alpha + 3) \cdot (3\alpha - 1) \\ &= 9\alpha^2 + 6\alpha - 1 \\ &> 9\alpha^2 + 5\alpha \\ &= \alpha \cdot (3\alpha - 1 + (3\alpha + 3) \cdot 2) \\ &= \alpha \cdot f_{x_3}(3\alpha + 3). \end{aligned}$$

Thus, neither $\{x_1\}$ nor $\{x_3\}$ is an α -approximation and $\{x_1, x_3\}$ is the only α -approximation that consists only of supported solutions. However, $\{x_2\}$ is an α -approximation: For any $\lambda \in [0, \infty)$, we have

$$f_{x_2}(\lambda) = 2\alpha + \lambda \cdot 2\alpha = \alpha \cdot (2 + 2\lambda) \leq \alpha \cdot (2 + (3\alpha - 1) \cdot \lambda) = \alpha \cdot f_{x_1}(\lambda)$$

and

$$f_{x_2}(\lambda) = 2\alpha + \lambda \cdot 2\alpha = \alpha \cdot (2 + 2\lambda) \leq \alpha \cdot (3\alpha - 1 + 2\lambda) = \alpha \cdot f_{x_3}(\lambda).$$

We, thus, have the following Corollary:

Corollary 4 *There exist instances of linear parametric optimization problems in which no α -approximation S consisting only of supported solutions has cardinality $|S| < 2 \cdot |S^*|$, where S^* is a minimum-cardinality α -approximation.*

5 Applications

In this section, we show that our general results apply to the parametric versions of many well-known, classical optimization problems including the parametric shortest path problem, the parametric assignment problem, a general class of parametric mixed integer linear programs that includes the parametric minimum cost flow problem, and the parametric metric traveling salesman problem. As will be discussed below, for each of these parametric problems, either the number of breakpoints in the optimal value function can be super-polynomial, which implies that solving the parametric problem exactly requires the generation of a super-polynomial number of solutions, or the corresponding non-parametric version is already NP-hard.

Parametric shortest path problem In the single-pair version of the parametric shortest path problem, we are given a directed graph $G = (V, R)$, where $|V| = n$ and $|R| = m$, together with a source node $s \in V$ and a destination node $t \in V$, where $s \neq t$. Each arc $r \in R$ has a parametric length of the form $a_r + \lambda \cdot b_r$, where $a_r, b_r \in \mathbb{N}_0$ are non-negative integers. The goal is to compute an s - t -path P_λ of minimum total length $\sum_{r \in P_\lambda} (a_r + \lambda \cdot b_r)$ for each $\lambda \geq 0$.

Since the arc lengths $a_r + \lambda \cdot b_r$ are non-negative for each $\lambda \geq 0$, one can restrict to *simple* s - t -paths as feasible solutions, and an upper bound UB as required in Algorithm 1 is given by summing up the $n - 1$ largest values a_r and summing up the $n - 1$ largest values b_r and taking the maximum of these two sums, which can easily be computed in polynomial time. The non-parametric problem can be solved in polynomial time $\mathcal{O}(m + n \log n)$ for any fixed $\lambda \geq 0$ by Dijkstra's algorithm, where $n = |V|$ and $m = |R|$ (see, e.g., Schrijver 2003). Hence, Corollary 1 and Remark 2 yield an FPTAS with running time $\mathcal{O}\left(\frac{1}{\varepsilon} \cdot (m + n \log n) \cdot \log nC\right)$, where C denotes the maximum among all values a_r, b_r . Moreover, by Corollary 3, we obtain an approximation of cardinality at most twice the minimum possible.

On the other hand, the number of breakpoints in the optimal value function is at least $n^{\Omega(\log n)}$ in the worst case even under our assumptions of non-negative, integer values a_r, b_r and for $\lambda \in \mathbb{R}_{\geq 0}$ (Carstensen 1983b; Nikolova et al. 2006).

Parametric assignment problem In the parametric assignment problem, we are given a bipartite, undirected graph $G = (U, V, E)$ with $|U| = |V| = n$ and $|E| = m$. Each edge $e \in E$ has a parametric weight of the form $a_e + \lambda \cdot b_e$, where $a_e, b_e \in \mathbb{N}_0$ are non-negative integers. The goal is to compute an assignment A_λ of minimum total weight $\sum_{e \in A_\lambda} (a_e + \lambda \cdot b_e)$ for each $\lambda \geq 0$.

Similar to the parametric shortest path problem, an upper bound UB as required in Algorithm 1 is given by summing up the n largest values a_r and summing up the n largest values b_r and taking the maximum of these two sums. The non-parametric problem can be solved in polynomial time $\mathcal{O}(n^3)$ for any fixed value $\lambda \geq 0$ (see, e.g., Schrijver 2003). Hence, Corollary 1 and Remark 2 yield an FPTAS with running time $\mathcal{O}\left(\frac{1}{\varepsilon} \cdot n^3 \cdot \log nC\right)$, where C denotes the maximum among all values a_e, b_e . Moreover, Corollary 3 yields an additional bound on the cardinality of the computed approximation.

On the other hand, applying the well-known transformation from the shortest s - t -path problem to the assignment problem (see, e.g., Lawler 2001) to the instances of the shortest s - t -path problem with super-polynomially many breakpoints presented in Carstensen (1983b), Nikolova et al. (2006) shows that the number of breakpoints in the parametric assignment problem can be super-polynomial as well (see also Gassner and Klinz 2010).

Parametric MIPs over integral polytopes A very general class of problems our results can be applied to are parametric mixed integer linear programs (parametric MIPs) with non-negative, integer objective function coefficients whose feasible set is of the form $P \cap (\mathbb{Z}^p \times \mathbb{R}^{n-p})$, where $P \subseteq \mathbb{R}_{\geq 0}^n$ is an integral polytope. More formally, consider a parametric MIP of the form

$$\begin{aligned}
 & \min / \max (a + \lambda b)^\top x \\
 & \text{s.t. } Ax = d \\
 & \quad Bx \leq e \\
 & \quad x \geq 0 \\
 & \quad x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}
 \end{aligned}$$

where A, B are rational matrices with n rows, d, e are rational vectors of the appropriate length, and $a, b \in \mathbb{N}_0^n$ are non-negative, rational vectors. We assume that the polyhedron $P := \{x \in \mathbb{R}^n : Ax = d, Bx \leq e, x \geq 0\} \subseteq \mathbb{R}^n$ is an integral polytope, i.e., it is bounded and each of its (finitely many) extreme points is an integral point.

Since, for each $\lambda \geq 0$, there exists an extreme point of P that is optimal for the non-parametric problem Π_λ , one can restrict to the extreme points when solving the problem. Since $\bar{x} \in \mathbb{N}_0^n$ for each extreme point \bar{x} of P and since $a, b \in \mathbb{N}_0^n$, the values $a(\bar{x}) = a^\top \bar{x}$ and $b(\bar{x}) = b^\top \bar{x}$ are non-negative integers. In order to solve the non-parametric problem for any fixed value $\lambda \geq 0$, we can simply solve the linear programming relaxation $\min / \max\{(a + \lambda b)^\top x : x \in P\}$ in polynomial time. This yields an optimal extreme point of P , which is integral by our assumptions. Similarly, an upper bound UB as required in Algorithm 1 can be computed in polynomial time by solving the two linear programs $\max\{a^\top x : x \in P\}$ and $\max\{b^\top x : x \in P\}$, and taking the maximum of the two resulting (integral) optimal objective values.

While Corollary 1 yields an FPTAS for any parametric MIP as above, it is well known that the number of breakpoints in the optimal value function can be exponential in the number n of variables (Carstensen 1983a; Murty 1980).

An important parametric optimization problem that can be viewed as a special case of a parametric MIP as above is the parametric minimum cost flow problem, in which we are given a directed graph $G = (V, R)$ together with a source node $s \in V$ and a destination node $t \in V$, where $s \neq t$, and an integral desired flow value $F \in \mathbb{N}_0$. Each arc $r \in R$ has an integral capacity $u_r \in \mathbb{N}_0$ and a parametric cost of the form $a_r + \lambda \cdot b_r$, where $a_r, b_r \in \mathbb{N}_0$ are non-negative integers. The goal is to compute a feasible s - t -flow x with flow value F of minimum total cost $\sum_{r \in R} (a_r + \lambda \cdot b_r) \cdot x_r$ for each $\lambda \geq 0$. Here, a large variety of (strongly) polynomial algorithms exist for the non-parametric problem, see, e.g., Ahuja et al. (1993). An upper bound UB can either be obtained by solving two linear programs as above, or by taking the maximum of $\sum_{r \in R} a_r \cdot u_r$ and $\sum_{r \in R} b_r \cdot u_r$. Using the latter and applying the enhanced capacity scaling algorithm to solve the non-parametric problem, which runs in $\mathcal{O}((m \cdot \log n)(m + n \cdot \log n))$ time on a graph with n nodes and m arcs (Ahuja et al. 1993), Corollary 1 and Remark 2 yield an FPTAS with running time $\mathcal{O}(\frac{1}{\epsilon} \cdot (m \cdot \log n)(m + n \cdot \log n) \cdot \log mCU)$, where C denotes the maximum among all values a_r, b_r , and $U := \max_{r \in R} u_r$. Corollary 3, again, lets us bound the cardinality of the approximation.

On the other hand, the optimal value function can have $\Omega(2^n)$ breakpoints even under our assumptions of non-negative, integer values a_r, b_r and for $\lambda \in \mathbb{R}_{\geq 0}$ (Ruhe 1988).

Parametric metric traveling salesman problem In the parametric version of the (symmetric) metric traveling salesman problem (TSP), we are given a complete undirected

graph $G = (V, E)$, where $|V| = n$. Each edge $e \in E = V \times V$ has a parametric length of the form $a_e + \lambda \cdot b_e$, where $a_e, b_e \in \mathbb{N}_0$ are non-negative integers that satisfy the triangle inequality: For any three vertices $u, v, w \in V$, we have $a(u, w) \leq a(u, v) + a(v, w)$ and $b(u, w) \leq b(u, v) + b(v, w)$. The goal is to compute a Hamiltonian cycle H_λ of minimum total length $\sum_{e \in H_\lambda} (a_e + \lambda \cdot b_e)$ for each $\lambda \geq 0$.

Similar to, e.g., the shortest path problem, an upper bound UB can be obtained by taking the maximum of the sum of the n largest values a_e and the sum of the n largest values b_e . The non-parametric metric TSP can be $\frac{3}{2}$ -approximated in $\mathcal{O}(n^3)$ time using Christofides' algorithm (Christofides 1976). Hence, Remark 2 yields a $(\frac{3}{2} + \varepsilon)$ -approximation algorithm whose running time is in $\mathcal{O}(\frac{1}{\varepsilon} \cdot n^3 \cdot (\log nC))$, where C denotes the maximum among all values a_e, b_e .

On the other hand, it is well-known that even the non-parametric metric TSP is APX-complete (see, e.g., Papadimitriou and Yannakakis 1993).

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows. Prentice Hall, Englewood Cliffs
- Arai T, Ueno S, Kajitani Y (1993) Generalization of a theorem on the parametric maximum flow problem. *Discrete Appl Math* 41(1):69–74
- Bazgan C, Herzel A, Ruzika S, Thielen C, Vanderpooten D (2019) An FPTAS for a general class of parametric optimization problems. In: Proceedings of the 25th international computing and combinatorics conference (COCOON), LNCS, vol 11653, pp 25–37
- Carstensen PJ (1983a) Complexity of some parametric integer and network programming problems. *Math Program* 26(1):64–75
- Carstensen PJ (1983b) The complexity of some problems in parametric, linear, and combinatorial programming. Ph.D. thesis, University of Michigan
- Christofides N (1976) Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh
- Daskalakis C, Diakonikolas I, Yannakakis M (2016) How good is the chord algorithm? *SIAM J Comput* 45(3):811–858
- Diakonikolas I (2011) Approximation of multiobjective optimization problems. Ph.D. thesis, Columbia University
- Diakonikolas I, Yannakakis M (2008) Succinct approximate convex Pareto curves. In: Proceedings of the 19th annual ACM-SIAM symposium on discrete algorithms (SODA), pp 74–83. SIAM
- Eisner MJ, Severance DG (1976) Mathematical techniques for efficient record segmentation in large shared databases. *J ACM* 23(4):619–635
- Fernández-Baca D, Slutzki G, Eppstein D (1996) Using sparsification for parametric minimum spanning tree problems. In: Proceedings of the 5th Scandinavian Workshop on Algorithm Theory (SWAT), LNCS, vol 1097, pp 149–160
- Gallo G, Grigoriadis MD, Tarjan RE (1989) A fast parametric maximum flow algorithm and applications. *SIAM J Comput* 18(1):30–55

- Gassner E, Klinz B (2010) A fast parametric assignment algorithm with applications in max-algebra. *Networks* 55(2):61–77
- Giudici A, Halfmann P, Ruzika S, Thielen C (2017) Approximation schemes for the parametric knapsack problem. *Inf Process Lett* 120:11–15
- Graves SC, Orlin JB (1985) A minimum concave-cost dynamic network flow problem with an application to lot-sizing. *Networks* 15:59–71
- Halman N, Holzhauser M, Krumke SO (2018) An FPTAS for the knapsack problem with parametric weights. *Oper Res Lett* 46(5):487–491
- Holzhauser M, Krumke SO (2017) An FPTAS for the parametric knapsack problem. *Inf Process Lett* 126:43–47
- Karp RM, Orlin JB (1981) Parametric shortest path algorithms with an application to cyclic staffing. *Discrete Appl Math* 3(1):37–45
- Lawler E (2001) *Combinatorial optimization: networks and matroids*. Dover Publications, New York
- McCormick ST (1999) Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. *Oper Res* 47(5):744–756
- Mitsos A, Barton PI (2009) Parametric mixed-integer 0–1 linear programming: the general case for a single parameter. *Eur J Oper Res* 194(3):663–686
- Mulmuley K, Shah P (2001) A lower bound for the shortest path problem. *J Comput Syst Sci* 63(2):253–267
- Murty K (1980) Computational complexity of parametric linear programming. *Math Program* 19(1):213–219
- Nikolova E, Kelner JA, Brand M, Mitzenmacher M (2006) Stochastic shortest paths via quasi-convex maximization. In: *Proceedings of the 14th annual European symposium on algorithms (ESA), LNCS*, vol 4168, pp 552–563
- Orlin JB, Rothblum UG (1985) Computing optimal scalings by parametric network algorithms. *Math Program* 32:1–10
- Papadimitriou C, Yannakakis M (1993) The traveling salesman problem with distances one and two. *Math Oper Res* 18(1):1–11
- Ruhe G (1988) Complexity results for multicriterial and parametric network flows using a pathological graph of Zadeh. *Z Oper Res* 32(1):9–27
- Schneider H, Schneider MH (1991) Max-balancing weighted directed graphs and matrix scaling. *Math Oper Res* 16(1):208–222
- Schrijver A (2003) *Combinatorial optimization. Algorithms and combinatorics*, vol 24. Springer, Heidelberg
- Scutellà MG (2007) A note on the parametric maximum flow problem and some related reoptimization issues. *Ann Oper Res* 150(1):231–244
- Young NE, Tarjan RE, Orlin JB (2006) Faster parametric shortest path and minimum-balance algorithms. *Networks* 21(2):205–221

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Cristina Bazgan¹ · Arne Herzel²  · Stefan Ruzika² · Clemens Thielen³ · Daniel Vanderpooten¹

✉ Arne Herzel
herzel@mathematik.uni-kl.de

Cristina Bazgan
bazgan@lamsade.dauphine.fr

Stefan Ruzika
ruzika@mathematik.uni-kl.de

Clemens Thielen
clemens.thielen@tum.de

Daniel Vanderpooten
daniel.vanderpooten@lamsade.dauphine.fr

- ¹ PSL Research University, CNRS, LAMSADE, Université Paris-Dauphine, 75016 Paris, France
- ² Department of Mathematics, University of Kaiserslautern, Paul-Ehrlich-Str. 14, 67663 Kaiserslautern, Germany
- ³ TUM Campus Straubing for Biotechnology and Sustainability, Technical University of Munich, Am Essigberg 3, 94315 Straubing, Germany