An FPTAS for a General Class of Parametric Optimization Problems

Cristina Bazgan[1], Arne Herzel[2(✉)], Stefan Ruzika[2], Clemens Thielen[2], and Daniel Vanderpooten[1]

[1] Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, 75016 Paris, France
{bazgan,vanderpooten}@lamsade.dauphine.fr

[2] Department of Mathematics, University of Kaiserslautern, Paul-Ehrlich-Str. 14, 67663 Kaiserslautern, Germany
{herzel,ruzika,thielen}@mathematik.uni-kl.de

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 (parametric) fully-polynomial time approximation scheme for a general class of parametric optimization problems for which (i) the parameter varies on the nonnegative real line, (ii) the non-parametric problem is solvable in polynomial time, and (iii) the slopes and intercepts of the value functions of the feasible solutions are nonnegative, integer values below a polynomial-time computable upper bound. In particular, under mild assumptions, we obtain the first parametric FPTAS for each of the specific problems mentioned above.

Keywords: Parametric optimization · Approximation scheme · Parametric assignment problem · Parametric minimum-cost flow problem · Parametric shortest path problem

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.

D.-Z. Du et al. (Eds.): COCOON 2019, LNCS 11653, pp. 25–37, 2019.
https://doi.org/10.1007/978-3-030-26176-4_3

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 $(-\infty, \lambda_1], [\lambda_1, \lambda_2], \ldots, [\lambda_{K-1}, \lambda_K], [\lambda_K, +\infty)$ 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 $\lambda_1, \ldots, \lambda_K$ (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. Prominent examples include the parametric shortest path problem [4,13,17,23], the parametric assignment problem [8], and the parametric minimum cost flow problem [3]. 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 [16] 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 [5], which requires to solve $\mathcal{O}(K)$ non-parametric problems for fixed values of the parameter.

Carstensen [3] 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 $\{-M, \ldots, M\}$ 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 [6] or several special cases of the parametric maximum flow problem [2,7,15,22] 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., [4,19] for the parametric shortest path problem, [8] for the

parametric assignment problem, and [20] for the parametric minimum cost flow problem. This, in particular, implies that there cannot exist any polynomial-time algorithm for these problems even if $\mathsf{P} = \mathsf{NP}$, which provides a strong motivation for the design of approximation algorithms.

So far, only very few approximation algorithms are known for parametric optimization problems. Approximation schemes for the parametric version of the 0-1 knapsack problem have been presented in [9,12], and an approximation 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 [10]. Moreover, an approximation scheme for a class of parametric discrete optimization problems on graphs, whose technique could potentially be generalized to further problems, has been proposed in [11].

Our Contribution. We provide a (parametric) fully-polynomial time approximation scheme (FPTAS) for a general class of parametric optimization problems. This means that, for any problem from this class and any given $\varepsilon > 0$, there exists an algorithm with running time polynomial in the input size and $1/\varepsilon$ that computes a partition of the set $\mathbb{R}_{\geq 0}$ of possible parameter values into polynomially many intervals together with a solution for each interval that $(1+\varepsilon)$-approximates all feasible solutions for all values of λ within the interval.

Our FPTAS can be viewed as an approximate version of the well-known Eisner-Severance method for parametric optimization problems [5]. It applies to all parametric optimization problems for which the parameter varies on the nonnegative real line, the non-parametric problem is solvable 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. In particular, 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 when presenting the applications of our method in Sect. 4, 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.

2 Preliminaries

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

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

We assume that the functions $a, b : X \to \mathbb{N}_0$ defining the intercepts and slopes of the value functions, respectively, take only nonnegative integer values and are polynomial-time computable. Moreover, we assume that we can compute

a rational upper bound UB such that $a(x), b(x) \leq \text{UB}$ for all $x \in X$ in polynomial time. In particular, this implies that UB is of polynomial encoding length.[1]

For any fixed value $\lambda \geq 0$, we let Π_λ denote the non-parametric problem obtained from Π by fixing the parameter value to λ. We assume that, for each $\lambda \geq 0$, this non-parametric problem Π_λ can be solved exactly in polynomial time by an algorithm ALG. The running time of ALG will be denoted by T_{ALG}, where we assume that this running time is at least as large as the time needed in order to compute the objective value $f_\lambda(x) = a(x) + \lambda \cdot b(x)$ of any feasible solution x of Π_λ.[2]

The above assumptions directly imply that the optimal value function mapping $\lambda \in \mathbb{R}_{\geq 0}$ to the optimal objective value of the non-parametric problem Π_λ is piecewise linear, increasing, and concave (for minimization problems) or convex (for maximization problems): Since there are at most $\text{UB} + 1$ possible integer values for each of $a(x)$ and $b(x)$, the function mapping λ to the objective value $f_\lambda(x) = a(x) + \lambda \cdot b(x)$ of a given feasible solution $x \in X$ is one of at most $(\text{UB}+1)^2$ many different affine functions. Consequently, the optimal value function given as $\lambda \mapsto \min / \max\{f_\lambda(x) : x \in X\}$ is the minimum/maximum of finitely many affine functions. The finitely many values of λ at which the slope of the optimal value function (or, equivalently, the set of optimal solutions of Π_λ) changes are called *breakpoints* of the optimal value function.

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.

Definition 1. *For $\alpha \geq 1$, an* α-approximation $(I^1, x^1), \ldots, (I^k, x^k)$ *for an instance of a parametric optimization problem Π consists of a cover of $\mathbb{R}_{\geq 0}$ by finitely many intervals $I^1, \ldots, I^k$ together with corresponding feasible solutions $x^1, \ldots, x^k$ such that, for each $j \in \{1, \ldots, k\}$, the solution x^j is an α-approximation for the corresponding instance of the non-parametric problem Π_λ for all values $\lambda \in I^j$, i.e.,*

$$f_\lambda(x^j) \leq \alpha \cdot f_\lambda(x) \text{ for all } x \in X \text{ and all } \lambda \in I^j.$$

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*

[1] Note that, the *numerical value* of UB can still be *exponential* in the input size of the problem, so there can still exist exponentially many different slopes $b(x)$ and intercepts $a(x)$.

[2] 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}. If the assumption is removed, our results still hold when replacing T_{ALG} in the running time of our algorithm by the maximum of T_{ALG} and the time needed for computing any value $f_\lambda(x)$.

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 An FPTAS

We now present our FPTAS for the general parametric optimization problem (1). To this end, we first describe the general functioning of the algorithm before formally stating and proving several auxiliary results needed for proving its correctness and analyzing its running time.

The algorithm, which is formally stated in Algorithm 1, starts by computing an upper bound UB on the values of $a(\cdot)$ and $b(\cdot)$, which is possible in polynomial time by our assumptions on the problem. It then starts with the initial parameter interval $[\lambda_{\min}, \lambda_{\max}]$, where $\lambda_{\min} := \frac{1}{\mathrm{UB}+1}$ is chosen such that an optimal solution $x^{\min}$ of the non-parametric problem $\Pi_{\lambda_{\min}}$ is optimal for Π_λ for all $\lambda \in [0, \lambda_{\min}]$ and $\lambda_{\max} := \mathrm{UB} + 1$ is chosen such that an optimal solution $x^{\max}$ of the non-parametric problem $\Pi_{\lambda_{\max}}$ is optimal for Π_λ for all $\lambda \in [\lambda_{\max}, +\infty)$ (see Lemma 3).

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 $[\lambda_{\min}, \lambda_{\max}]$ and optimal solutions x^ℓ, x^r of the respective non-parametric problems $\Pi_{\lambda_\ell}, \Pi_{\lambda_r}$ at the interval boundaries. The queue is initialized as $Q = \{([\lambda_{\min}, \lambda_{\max}], x^{\min}, x^{\max})\}$, where $x^{\min}, x^{\max}$ are optimal for $\Pi_{\lambda_{\min}}, \Pi_{\lambda_{\max}}$, respectively.

Afterwards, in each iteration, the algorithm extracts an element $([\lambda_\ell, \lambda_r], x^\ell, x^r)$ from the queue and checks whether one of the two solutions x^ℓ, x^r obtained at the boundaries of the parameter interval $[\lambda_\ell, \lambda_r]$ is a $(1+\varepsilon)$-approximate solution also at the other boundary of the interval. In this case, Lemma 1 below implies that this boundary solution is a $(1+\varepsilon)$-approximate solution within the whole interval $[\lambda_\ell, \lambda_r]$ and the pair consisting of the interval $[\lambda_\ell, \lambda_r]$ and this $(1+\varepsilon)$-approximate solution is added to the solution set S. Otherwise, $[\lambda_\ell, \lambda_r]$ is bisected into the two subintervals $[\lambda_\ell, \lambda_m]$ and $[\lambda_m, \lambda_r]$, where $\lambda_m := \sqrt{\lambda_\ell \cdot \lambda_r}$ is the geometric mean of λ_ℓ and λ_r. This means that an optimal solution x^m of Π_{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.

This iterative subdivision of the initial parameter interval $[\lambda_{\min}, \lambda_{\max}]$ can be viewed as creating a binary tree: the root corresponds to the initialization, in which the two non-parametric problems $\Pi_{\lambda_{\min}}$ and $\Pi_{\lambda_{\max}}$ are solved. Each other internal node of the tree corresponds to an interval $[\lambda_\ell, \lambda_r]$ that is further subdivided into $[\lambda_\ell, \lambda_m]$ and $[\lambda_m, \lambda_r]$, which requires the solution of one non-parametric problem Π_{λ_m}. In order to bound the total number of non-parametric problems solved within the algorithm, it is, thus, sufficient to upper bound the number of nodes in this binary tree.

We now prove the auxiliary results mentioned in the algorithm description above. The first lemma shows that a solution that is optimal at one boundary of a parameter interval and simultaneously α-approximate at the other interval boundary must be α-approximate within the whole interval.

Algorithm 1. An FPTAS for parametric optimization problems

input : an instance of a parametric optimization problem Π as in (1), $\varepsilon > 0$, an exact algorithm ALG for the non-parametric version of Π
output: a $(1+\varepsilon)$-approximation for Π

Compute an upper bound UB such that $a(x), b(x) \leq \text{UB}$ for all $x \in X$
$\lambda_{\min} \leftarrow \frac{1}{\text{UB}+1}$; $\lambda_{\max} \leftarrow \text{UB} + 1$
$x^{\min} \leftarrow \text{ALG}(\lambda_{\min})$; $x^{\max} \leftarrow \text{ALG}(\lambda_{\max})$
$Q \leftarrow \{([\lambda_{\min}, \lambda_{\max}], x^{\min}, x^{\max})\}$ `/* queue of intervals still to be considered */`
$S \leftarrow \{([0, \lambda_{\min}], x^{\min}), ([\lambda_{\max}, +\infty), x^{\max})\}$ `/* solution set */`
while $Q \neq \emptyset$ **do**
    Extract some element $([\lambda_\ell, \lambda_r], x^\ell, x^r)$ from Q
    if $f_{\lambda_\ell}(x^r) \leq (1+\varepsilon) \cdot f_{\lambda_\ell}(x^\ell)$ **then**
        $S \leftarrow S \cup \{([\lambda_\ell, \lambda_r], x^r)\}$
    else if $f_{\lambda_r}(x^\ell) \leq (1+\varepsilon) \cdot f_{\lambda_r}(x^r)$ **then**
        $S \leftarrow S \cup \{([\lambda_\ell, \lambda_r], x^\ell)\}$
    else
        $\lambda_m \leftarrow \sqrt{\lambda_\ell \cdot \lambda_r}$
        $x^m \leftarrow \text{ALG}(\lambda_m)$
        $Q \leftarrow Q \cup \{([\lambda_\ell, \lambda_m], x^\ell, x^m), ([\lambda_m, \lambda_r], x^m, x^r)\}$
return S

Lemma 1. *Let $[\underline{\lambda}, \bar{\lambda}] \subset \mathbb{R}_{\geq 0}$ be an interval, and let $\underline{x}, \bar{x} \in X$ be optimal solutions of $\Pi_{\underline{\lambda}}$ and $\Pi_{\bar{\lambda}}$, respectively. Then, for any $\alpha \geq 1$:*

(1) If $f_{\bar{\lambda}}(\underline{x}) \leq \alpha \cdot f_{\bar{\lambda}}(\bar{x})$, then $f_\lambda(\underline{x}) \leq \alpha \cdot f_\lambda(x)$ for all $x \in X$ and all $\lambda \in [\underline{\lambda}, \bar{\lambda}]$.
(2) If $f_{\underline{\lambda}}(\bar{x}) \leq \alpha \cdot f_{\underline{\lambda}}(\underline{x})$, then $f_\lambda(\bar{x}) \leq \alpha \cdot f_\lambda(x)$ for all $x \in X$ and all $\lambda \in [\underline{\lambda}, \bar{\lambda}]$.

Proof. We only prove (1) – the proof of (2) is analogous.

Let $f_{\bar{\lambda}}(\underline{x}) \leq \alpha \cdot f_{\bar{\lambda}}(\bar{x})$, i.e., $a(\underline{x}) + \bar{\lambda} b(\underline{x}) \leq \alpha \cdot (a(\bar{x}) + \bar{\lambda} b(\bar{x}))$. Fix some $\lambda \in [\underline{\lambda}, \bar{\lambda}]$. Then $\lambda = \gamma \underline{\lambda} + (1-\gamma)\bar{\lambda}$ for some $\gamma \in [0,1]$ and, for each $x \in X$, we have:

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

Here, the first inequality follows by the assumption of (1), and the second inequality follows since $\underline{x}, \bar{x}$ are optimal solutions of $\Pi_{\underline{\lambda}}$ and $\Pi_{\bar{\lambda}}$, respectively. □

The following lemma shows that a solution that is optimal for some value λ^* of the parameter is always $(1+\varepsilon)$-approximate in a neighborhood of λ^*.

Lemma 2. *For $\lambda^* \in [0, \infty)$ and $\varepsilon > 0$, let $\underline{\lambda} = \frac{1}{1+\varepsilon} \cdot \lambda^*$ and $\bar{\lambda} = (1+\varepsilon) \cdot \lambda^*$. Also, let x^* be an optimal solution for Π_{λ^*}. Then $f_\lambda(x^*) \leq (1+\varepsilon) \cdot f_\lambda(x)$ for all $x \in X$ and all $\lambda \in [\underline{\lambda}, \bar{\lambda}]$.*

Proof. First, note that x^* is $(1+\varepsilon)$-approximate for $\Pi_{\bar{\lambda}}$: Let $\bar{x}$ be an optimal solution for $\Pi_{\bar{\lambda}}$. Then we have

$$\begin{aligned} f_{\bar{\lambda}}(x^*) &= a(x^*) + \bar{\lambda} \cdot b(x^*) \\ &= a(x^*) + (1+\varepsilon) \cdot \lambda^* \cdot b(x^*) \\ &\leq (1+\varepsilon) \cdot (a(x^*) + \lambda^* \cdot b(x^*)) \\ &= (1+\varepsilon) \cdot f_{\lambda^*}(x^*) \\ &\leq (1+\varepsilon) \cdot f_{\bar{\lambda}}(\bar{x}), \end{aligned}$$

where the last inequality is due to the monotonicity of the optimal cost curve. Moreover, x^* is $(1+\varepsilon)$-approximate for $\Pi_{\underline{\lambda}}$: Let $\underline{x}$ be an optimal solution for $\Pi_{\underline{\lambda}}$ and let x^0 be an optimal solution for Π_0. Then, since $\underline{\lambda} = \frac{1}{1+\varepsilon} \cdot \lambda^* = \frac{1}{1+\varepsilon} \cdot \lambda^* + \frac{\varepsilon}{1+\varepsilon} \cdot 0$, we have

$$\begin{aligned} f_{\underline{\lambda}}(x^*) &\leq f_{\lambda^*}(x^*) \\ &\leq f_{\lambda^*}(x^*) + \varepsilon \cdot f_0(x^0) \\ &= (1+\varepsilon) \cdot \left(\frac{1}{1+\varepsilon} \cdot f_{\lambda^*}(x^*) + \frac{\varepsilon}{1+\varepsilon} \cdot f_0(x^0) \right) \\ &\leq (1+\varepsilon) \cdot f_{\underline{\lambda}}(\underline{x}), \end{aligned}$$

where the first inequality is due to the monotonicity of $\lambda \mapsto f_\lambda(x^*)$ and the last inequality is due to the concavity of the optimal cost curve. Now, the claim follows from applying Lemma 1. □

The next result justifies our choice of the initial parameter interval $[\lambda_{\min}, \lambda_{\max}]$.

Lemma 3. *Let $\lambda_{\min} := \frac{1}{\mathrm{UB}+1}$ and $\lambda_{\max} := \mathrm{UB} + 1$ as in Algorithm 1. Then the solution $x^{\min} = \mathrm{ALG}(\lambda_{\min})$ is optimal for Π_λ for all $\lambda \in [0, \lambda_{\min}]$ and the solution $x^{\max} = \mathrm{ALG}(\lambda_{\max})$ is optimal for Π_λ for all $\lambda \in [\lambda_{\max}, +\infty)$.*

Proof. Let $\lambda \in [\lambda_{\max}, +\infty)$ and $x \in X$ be an arbitrary solution. Since $x^{\max}$ is optimal for $\Pi_{\lambda_{\max}}$, we have $f_{\lambda_{\max}}(x^{\max}) \leq f_{\lambda_{\max}}(x)$, i.e., $a(x^{\max}) + (\mathrm{UB} + 1) \cdot b(x^{\max}) \leq a(x) + (\mathrm{UB} + 1) \cdot b(x)$. Reordering terms, and using that $0 \leq a(x), a(x^{\max}) \leq \mathrm{UB}$, we obtain that

$$b(x^{\max}) - b(x) \leq \frac{a(x) - a(x^{\max})}{\mathrm{UB} + 1} \leq \frac{\mathrm{UB}}{\mathrm{UB} + 1} < 1.$$

Since $b(x^{\max}) - b(x) \in \mathbb{Z}$ by integrality of the values of b, this implies that $b(x^{\max}) - b(x) \leq 0$, i.e., $b(x^{\max}) \leq b(x)$. Consequently, using that $x^{\max}$ is optimal

for $\Pi_{\lambda_{\max}}$, we obtain

$$\begin{aligned} f_\lambda(x^{\max}) &= \underbrace{f_{\lambda_{\max}}(x^{\max})}_{\leq f_{\lambda_{\max}}(x)} + \underbrace{(\lambda - \lambda_{\max})}_{\geq 0} \cdot \underbrace{b(x^{\max})}_{\leq b(x)} \\ &\leq f_{\lambda_{\max}}(x) + (\lambda - \lambda_{\max}) \cdot b(x) \\ &= f_\lambda(x). \end{aligned}$$

Since $x \in X$ was arbitrary, this proves the optimality of $x^{\max}$ for Π_λ.

Now, consider the interval $[0, \lambda_{\min}]$. We know that $x^{\min}$ is optimal for $\Pi_{\lambda_{\min}}$. Analogously to the above arguments, we can show that $a(x^{\min}) \leq a(x)$, i.e., $f_0(x^{\min}) \leq f_0(x)$, for all $x \in X$. Thus, for any $\lambda \in [0, \lambda_{\min}]$, we obtain $f_\lambda(x^{\min}) \leq f_\lambda(x)$ by applying Lemma 1. □

We are now ready to show that Algorithm 1 yields an FPTAS for the parametric optimization problem (1):

Theorem 1. *Algorithm 1 returns a* $(1+\varepsilon)$*-approximation of the parametric problem in time*

$$\mathcal{O}\left(T_{\mathrm{UB}} + T_{\mathrm{ALG}} \cdot \frac{1}{\varepsilon} \cdot \log \mathrm{UB}\right),$$

where T_{UB} *denotes the time needed for computing the upper bound* UB *and* T_{ALG} *denotes the running time of* ALG.

Proof. In order to prove the approximation guarantee, we first note that, at the beginning of each iteration of the while loop starting in line 6, the intervals corresponding to the first components of the elements of $Q \cup S$ form a cover of $\mathbb{R}_{\geq 0}$. Since $Q = \emptyset$ at termination, the approximation guarantee follows if we show that, for each element $(\hat{I}, \hat{x})$ in the final solution set S returned by the algorithm, we have $f_\lambda(\hat{x}) \leq (1+\varepsilon) \cdot f_\lambda(x)$ for all $x \in X$ and all $\lambda \in \hat{I}$.

Consider an arbitrary element $(\hat{I}, \hat{x}) \in S$. If $(\hat{I}, \hat{x}) = ([0, \lambda_{\min}], x^{\min})$ or $(\hat{I}, \hat{x}) = ([\lambda_{\max}, +\infty), x^{\max})$, then the solution $\hat{x}$ is optimal for Π_λ for all $\lambda \in \hat{I}$ by Lemma 3.

Otherwise, $\hat{I} = [\hat{\lambda}_\ell, \hat{\lambda}_r]$ and $(\hat{I}, \hat{x})$ was added to S within the while loop starting in line 6, i.e., either in line 9 or in line 11. In this case, the solution $\hat{x}$ must be optimal for $\Pi_{\hat{\lambda}_\ell}$ or for $\Pi_{\hat{\lambda}_r}$: Whenever an element $([\lambda_\ell, \lambda_r], x^\ell, x^r)$ is added to the queue Q, the solution x^ℓ is optimal for Π_{λ_ℓ} and the solution x^r is optimal for Π_{λ_r}. Consequently, whenever $([\lambda_\ell, \lambda_r], x^\ell)$ or $([\lambda_\ell, \lambda_r], x^r)$ is added to S, the contained solution is optimal for Π_{λ_ℓ} or for Π_{λ_r}, respectively. Thus, if $(\hat{I}, \hat{x}) = ([\hat{\lambda}_\ell, \hat{\lambda}_r], \hat{x})$ was added to S in line 9 or 11, the solution $\hat{x}$ satisfies $f_\lambda(\hat{x}) \leq (1+\varepsilon) \cdot f_\lambda(x)$ for all $x \in X$ and all $\lambda \in \hat{I}$ by the if-statement in the previous line of the algorithm and Lemma 1.

We prove now the bound on the running time. Starting with the initial interval $[0, \lambda_{\max}]$, the algorithm 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. This process of bisecting can be viewed as creating a full binary tree: the root corresponds to the initial element $([\lambda_{\min}, \lambda_{\max}], x^{\min}, x^{\max})$, for which the two non-parametric problems $\Pi_{\lambda_{\min}}$ and $\Pi_{\lambda_{\max}}$ are solved. Each other node corresponds to an element $([\lambda_\ell, \lambda_r], x^\ell, x^r)$ that is extracted from Q in some iteration of the while loop, where x^ℓ and x^r are optimal for Π_{λ_ℓ} and Π_{λ_r}, respectively. If the interval $[\lambda_\ell, \lambda_r]$ is not bisected into $[\lambda_\ell, \lambda_m]$ and $[\lambda_m, \lambda_r]$, the corresponding node is a leaf of the tree, for which no non-parametric problem is solved. Otherwise, the node corresponding to $([\lambda_\ell, \lambda_r], x^\ell, x^r)$ is an internal node, for which the optimal solution x^m of the non-parametric problem Π_{λ_m} with $\lambda_m = \sqrt{\lambda_\ell \cdot \lambda_r}$ is computed, and whose two children correspond to $([\lambda_\ell, \lambda_m], x^\ell, x^m)$ and $([\lambda_m, \lambda_r], x^m, x^r)$. In order to bound the total number of non-parametric problems solved within the algorithm, it is, thus, sufficient to bound the number of (internal) nodes in this full binary tree. This is done by bounding the height of the tree.

In order to bound the height of the tree, we observe that due to Lemma 2, the algorithm never bisects an interval $[\lambda_\ell, \lambda_r]$, for which $\lambda_\ell \leq (1+\varepsilon) \cdot \lambda_r$, i.e., for which the ratio between λ_r and λ_ℓ satisfies $\frac{\lambda_r}{\lambda_\ell} \leq 1+\varepsilon$. Also note that $\lambda_m = \sqrt{\lambda_\ell \cdot \lambda_r}$ and, thus, for any subdivision $\{[\lambda_\ell, \lambda_m], [\lambda_m, \lambda_r]\}$ of $[\lambda_\ell, \lambda_r]$ in the algorithm, we have

$$\frac{\lambda_m}{\lambda_\ell} = \frac{\lambda_r}{\lambda_m} = \sqrt{\frac{\lambda_r}{\lambda_\ell}} = \left(\frac{\lambda_r}{\lambda_\ell}\right)^{\frac{1}{2}}.$$

Hence, for both intervals $[\lambda_\ell, \lambda_m]$ and $[\lambda_m, \lambda_r]$ in the subdivision, the ratio between the upper and lower boundary equals the square root of the corresponding ratio of the previous interval $[\lambda_\ell, \lambda_r]$.

This implies that an interval $[\lambda_{\ell,k}, \lambda_{r,k}]$ resulting from k consecutive subdivisions of an interval $[\lambda_\ell, \lambda_r]$ satisfies

$$\frac{\lambda_{\ell,k}}{\lambda_{r,k}} = \left(\frac{\lambda_\ell}{\lambda_r}\right)^{\frac{1}{2^k}}.$$

Thus, starting from the initial interval $[\lambda_{\min}, \lambda_{\max}] = [\frac{1}{\mathrm{UB}+1}, \mathrm{UB}+1]$, for which the ratio is $\frac{\lambda_{\max}}{\lambda_{\min}} = (\mathrm{UB}+1)^2$, there can be at most $\left\lceil \log_2 \left(\frac{\log((\mathrm{UB}+1)^2)}{\log(1+\varepsilon)} \right) \right\rceil$ consecutive subdivisions until the ratio between the interval boundaries becomes less or equal to $1+\varepsilon$, which upper bounds the height of the tree by $\left\lceil \log_2 \left(\frac{\log((\mathrm{UB}+1)^2)}{\log(1+\varepsilon)} \right) \right\rceil + 1$.

Since any binary tree of height h has at most $2^{h-1}-1$ internal nodes, we obtain an upper bound of

$$2^{\left\lceil \log_2 \left(\frac{\log((\mathrm{UB}+1)^2)}{\log(1+\varepsilon)} \right) \right\rceil} - 1 \in \mathcal{O}\left(\frac{\log \mathrm{UB}}{\varepsilon}\right)$$

for the number of internal nodes of the tree generated by the algorithm.

Adding the time T_{UB} needed for computing the upper bound UB at the beginning of the algorithm, this proves the claimed bound on the running time. □

4 Applications

In this section, we show that our general result applies to the parametric versions of many well-known, classical optimization problems including the parametric shortest path problem, the parametric assignment problem, and a general class of parametric mixed integer linear programs that includes the parametric minimum cost flow problem. As will be discussed below, for each of these parametric problems, 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.

Parametric Shortest Path Problem. In the single-pair version of the parametric shortest path problem, 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 $s \neq t$. 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 nonnegative 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 nonnegative 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 Π_λ 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., [21]). Hence, Theorem 1 yields 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.

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 nonnegative, integer values a_r, b_r and for $\lambda \in \mathbb{R}_{\geq 0}$ [4,19].

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 nonnegative 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 Π_λ can be solved in polynomial time $\mathcal{O}(n^3)$ for any fixed value $\lambda \geq 0$ (see, e.g., [21]). Hence, Theorem 1 yields 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.

On the other hand, applying the well-known transformation from the shortest s-t-path problem to the assignment problem (see, e.g., [14]) to the instances of the shortest s-t-path problem with super-polynomially many breakpoints presented in [4,19] shows that the number of breakpoints in the parametric assignment problem can be super-polynomial as well (see also [8]).

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 nonnegative, integer objective function coefficients whose feasible set is of the form $P \cap (\mathbb{Z}^p \times \mathbb{R}^{n-p})$, where $P \subseteq \mathbb{R}^n_{\geq 0}$ is an integral polytope. More formally, consider a parametric MIP of the form

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

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 nonnegative, integer 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 $a(\bar{x}) = a^T\bar{x}$ and $b(\bar{x}) = b^T\bar{x}$ are nonnegative integers. In order to solve the non-parametric problem Π_λ for any fixed value $\lambda \geq 0$, we can simply solve the linear programming relaxation $\min / \max\{(a+\lambda b)^T x : x \in P\}$ 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 $\max\{a^T x : x \in P\}$ and $\max\{b^T x : x \in P\}$ and taking the maximum of the two resulting (integral) optimal objective values.

While Theorem 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 [3,18].

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 nonnegative 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., [1]. 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 [1], Theorem 1 yields an FPTAS with running time $\mathcal{O}\left(\frac{1}{\varepsilon} \cdot (m \cdot \log n)(m + n \cdot \log n) \cdot \log mCU\right)$, where C denotes the maximum among all values a_r, b_r, and $U := \max_{r\in R} u_r$.

On the other hand, the optimal value function can have $\Omega(2^n)$ breakpoints even under our assumptions of nonnegative, integer values a_r, b_r and for $\lambda \in \mathbb{R}_{\geq 0}$ [20].

Acknowledgments. We thank Sven O. Krumke for pointing out a possible improvement of the running time of a first version of our algorithm.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs (1993)
2. Arai, T., Ueno, S., Kajitani, Y.: Generalization of a theorem on the parametric maximum flow problem. Discrete Appl. Math. **41**(1), 69–74 (1993)
3. Carstensen, P.J.: Complexity of some parametric integer and network programming problems. Math. Program. **26**(1), 64–75 (1983)
4. Carstensen, P.J.: The complexity of some problems in parametric, linear, and combinatorial programming. Ph.D. thesis, University of Michigan (1983)
5. Eisner, M.J., Severance, D.G.: Mathematical techniques for efficient record segmentation in large shared databases. J. ACM **23**(4), 619–635 (1976)
6. Fernández-Baca, D., Slutzki, G., Eppstein, D.: Using sparsification for parametric minimum spanning tree problems. In: Karlsson, R., Lingas, A. (eds.) SWAT 1996. LNCS, vol. 1097, pp. 149–160. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61422-2_128
7. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**(1), 30–55 (1989)
8. Gassner, E., Klinz, B.: A fast parametric assignment algorithm with applications in max-algebra. Networks **55**(2), 61–77 (2010)
9. Giudici, A., Halffmann, P., Ruzika, S., Thielen, C.: Approximation schemes for the parametric knapsack problem. Inf. Process. Lett. **120**, 11–15 (2017)
10. Halman, N., Holzhauser, M., Krumke, S.O.: An FPTAS for the knapsack problem with parametric weights. Oper. Res. Lett. **46**(5), 487–491 (2018)
11. Hertrich, C.: A parametric view on robust graph problems. Bachelor's thesis, University of Kaiserslautern, August 2016
12. Holzhauser, M., Krumke, S.O.: An FPTAS for the parametric knapsack problem. Inf. Process. Lett. **126**, 43–47 (2017)
13. Karp, R.M., Orlin, J.B.: Parametric shortest path algorithms with an application to cyclic staffing. Discrete Appl. Math. **3**(1), 37–45 (1981)
14. Lawler, E.: Combinatorial Optimization: Networks and Matroids. Dover Publications, New York (2001)
15. McCormick, S.T.: Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. Oper. Res. **47**(5), 744–756 (1999)

16. Mitsos, A., Barton, P.I.: Parametric mixed-integer 0–1 linear programming: the general case for a single parameter. Eur. J. Oper. Res. **194**(3), 663–686 (2009)
17. Mulmuley, K., Shah, P.: A lower bound for the shortest path problem. J. Comput. Syst. Sci. **63**(2), 253–267 (2001)
18. Murty, K.: Computational complexity of parametric linear programming. Math. Program. **19**(1), 213–219 (1980)
19. Nikolova, E., Kelner, J.A., Brand, M., Mitzenmacher, M.: Stochastic shortest paths via quasi-convex maximization. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 552–563. Springer, Heidelberg (2006). https://doi.org/10.1007/11841036_50
20. Ruhe, G.: Complexity results for multicriterial and parametric network flows using a pathological graph of Zadeh. Zeitschrift für Oper. Res. **32**(1), 9–27 (1988)
21. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Algorithms and Combinatorics, vol. 24, 1st edn. Springer, Heidelberg (2003)
22. Scutellà, M.G.: A note on the parametric maximum flow problem and some related reoptimization issues. Ann. Oper. Res. **150**(1), 231–244 (2007)
23. Young, N.E., Tarjan, R.E., Orlin, J.B.: Faster parametric shortest path and minimum-balance algorithms. Networks **21**(2), 205–221 (2006)