Utilitarian Mechanism Design for Multi-Objective Optimization*

Fabrizio Grandoni† Piotr Krysta‡ Stefano Leonardi§ Carmine Ventre¶

Abstract

In a classic optimization problem the complete input data is known to the algorithm. This assumption may not be true anymore in optimization problems motivated by the Internet where part of the input data is private knowledge of independent *selfish* agents. The goal of *algorithmic mechanism design* is to provide (in polynomial time) a solution to the optimization problem and a set of incentives for the agents such that disclosing the input data is a dominant strategy for the agents. In case of NP-hard problems, the solution computed should also be a good approximation of the optimum.

In this paper we focus on mechanism design for *multi-objective* optimization problems, where we are given the main objective function, and a set of secondary objectives which are modeled via budget constraints. Multi-objective optimization is a natural setting for mechanism design as many economical choices ask for a compromise between different, partially conflicting, goals. Our main contribution is showing that two of the main tools for the design of approximation algorithms for multi-objective optimization problems, namely approximate Pareto curves and Lagrangian relaxation, can lead to truthful approximation schemes.

By exploiting the method of approximate Pareto curves, we devise truthful FPTASs for multi-objective optimization problems whose exact version admits a pseudo-polynomial-time algorithm, as for instance the multi-budgeted versions of *minimum spanning tree*, *shortest path*, *maximum (perfect) matching*, and *matroid intersection*. Our technique applies also to *multi-dimensional knapsack* and *multi-unit combinatorial auctions*. Our FPTASs compute a $(1+\epsilon)$-approximate solution violating each budget constraint by a factor $(1+\epsilon)$. For a relevant sub-class of the mentioned problems we also present a PTAS (not violating any constraint), which combines the approach above with a novel monotone way to guess the heaviest elements in the optimum solution.

Finally, we present a universally truthful Las Vegas PTAS for minimum spanning tree with a single budget constraint. This result is based on the Lagrangian relaxation method, in combination with our monotone guessing step and a random perturbation step (ensuring low expected running time in a way similar to the smoothed analysis of algorithms). All the mentioned results match the best known approximation ratios, which however are obtained by non-truthful algorithms.

1 Introduction

A multi-objective combinatorial optimization problem is characterized by a set $\mathcal{S} \subseteq 2^{\mathcal{U}}$ of feasible solutions defined over a ground set $\mathcal{U}$ of m elements, and a set of objective cost functions $\ell_i : \mathcal{U} \to \mathbb{Q}^+$, $i = 0, \ldots, k$. In this paper we assume $k = O(1)$. The aim is finding a solution $S \in \mathcal{S}$ optimizing $\ell_i(S) = \sum_{e \in S} \ell_i(e)$ for all i, where optimizing means maximizing or minimizing, depending on the objective. A typical way to deal with multiple objectives is turning all the objectives but one into budget constraints (see, e.g., [21] and references therein for related approaches). More formally, let $\mathcal{X}$ be the set of incidence vectors corresponding to $\mathcal{S}$, and $B_i \in \mathbb{Q}^+$, $i = 1, \ldots, k$, a set of budgets. We consider a problem $\mathcal{P}$ defined as:

$$\begin{aligned} best \quad & \sum_{e \in \mathcal{U}} \ell_0(e)\, x_e \\ s.t. \quad & x \in \mathcal{X} \\ & \sum_{e \in \mathcal{U}} \ell_i(e)\, x_e \succeq_i B_i \quad \text{for } i = 1, \ldots, k. \end{aligned} \tag{$\mathcal{P}$}$$

Here, $best \in \{\max, \min\}$ and $\succeq_i \in \{\geq, \leq\}$. We refer to cost functions different from $\ell_0(\cdot)$ as *lengths*, and let $\succeq_0 = \geq$ if $best = \max$, and $\succeq_0 = \leq$ otherwise. We also let

*This work has been supported by EPSRC grant EP/F069502/1 and by DFG grant Kr 2332/1-3 within Emmy Noether program. The fourth author is also supported by the European Union under IST FET Integrated Project AEOLUS (IST-015964).

†Università di Roma Tor Vergata, Dipartimento di Informatica, Sistemi e Produzione, via del Politecnico 1, 00133 Roma, Italy. Email: grandoni@disp.uniroma2.it.

‡University of Liverpool, Department of Computer Science, Ashton Building, Ashton Street, Liverpool L69 3BX, U.K. Email: p.krysta@liverpool.ac.uk

§Sapienza Università di Roma, Dipartimento di Informatica e Sistemistica, Via Ariosto 25 00185 Roma, Italy. Email: leon@dis.uniroma1.it

¶University of Liverpool, Department of Computer Science, Ashton Building, Ashton Street, Liverpool L69 3BX, U.K. Email: carmine.ventre@liverpool.ac.uk

$sign_i = +1$ for $\succeq_i = \geq$ and $sign_i = -1$ otherwise. For a relation $\succeq$, we say that $a \succ b$ if $a \succeq b$ and $a \neq b$, and use $\prec$ for $\not\succeq$ and $\preceq$ for $\not\succ$.

This framework naturally models network design problems with resource constraints. For example, in the *budgeted minimum spanning tree* problem (BMST) $\mathcal{U}$ is the set of edges of a graph $G = (V, E)$, $\mathcal{S}$ is the set of spanning trees of G, $best = \min$ and there is a unique budget constraint with $\succeq_1 = \leq$. Here $\ell_0(e)$ can be used to model, e.g, the cost of establishing a link on e, and $\ell_1(e)$ the delay on that edge. The *budgeted shortest path* problem (BSP) is defined analogously, where spanning trees are replaced by the paths between two given nodes. Alternatively, $(\mathcal{U}, \mathcal{S})$ might define the intersection of two matroids.

In this paper we study this family of optimization problems from a game theoretical point of view. We are given a set of selfish agents, where agent f controls element $f \in \mathcal{U}$. The *type* $\ell^f = (\ell_0(f), \ldots, \ell_k(f))$ is considered as private knowledge of f. Agent f can *lie* by declaring a type $\ell'^f \neq \ell^f$, with the constraint $\ell_i(f) \succeq_i \ell'_i(f)$ for $i = 1, \ldots, k$ (while $\ell'_0(f)$ is arbitrary). In the examples above, agent f might declare a larger delay on f than the smallest possible $\ell_1(f)$ in order to reduce her operational costs (while f is not able to offer a delay smaller than $\ell_1(f)$). Moreover, f might declare a cost larger than $\ell_0(f)$ to maximize her revenue, or a smaller cost to maximize her chances to be part of the solution. (For further motivations on this model of selfish agents please refer to Section 2.)

The main goal of (utilitarian) mechanism design is forcing f to declare her true type. This is achieved by defining a set of *payments* $p^f(\cdot)$, $f \in \mathcal{U}$. Payments, type and computed solution define agents' *utility* (please see Section 2 for utility definition). Payments should be defined such that saying the truth is a *dominant strategy*, i.e. each agent maximizes her utility by declaring her true type, independently of what the other agents declare. In that case the mechanism is *truthful*. At the same time, we wish to design mechanisms which compute (efficiently) a solution approximating the optimum in a standard sense, with respect to the declarations. The objective of $\mathcal{P}$ is also called *social welfare* in this context.

1.1 Related work A classical technique to design truthful mechanisms is the Vickrey-Clarke-Groves (VCG) mechanism [24, 7, 12] which however requires to solve optimally the underlying optimization problem. For NP-hard problems one can use the approach of [5, 15], who show that, in order to obtain truthfulness, it is sufficient to design an approximation algorithm which is also monotone. Informally, an algorithm is monotone if, assuming that it computes a solution containing an element f for a given problem, then it computes a solution containing f also in the case that we replace, for some $i \in \{0, 1, \ldots, k\}$, $\ell_i(f)$ with $\bar{\ell}_i(f) \succeq_i \ell_i(f)$[1].

Efficient monotone algorithms are known only for a few special cases of the framework above. For example by setting $best = \max$, $\succeq_i = \leq$ for all $i \geq 1$, and using a trivial set of feasible solutions $\mathcal{S} = 2^{\mathcal{U}}$, one can model the *multi-dimensional knapsack* problem. For this problem a monotone $O(k^{1/B})$ approximation is given in [5], where B is the smallest budget. Their result extends to *multi-unit combinatorial auctions* for unknown single-minded bidders. (See [3, 8, 9, 14] for results on more general types of bidders). A monotone PTAS is known for *multiple knapsack* [5], which can be modeled in our framework by letting one agent control a constant number of elements. This extends to *multi-unit unit-demand auctions* for unknown multi-minded bidders with the same valuation for each alternative. For graph problems, monotone FPTASs are known for BSP and for BMST in the special case of bounded-treewidth graphs [5]. We note that these FPTASs do not violate the budget constraint, but it is not clear how to extend them to more than one budget. Another related result is a bicriteria monotone algorithm for the spanning arborescence problem [4].

1.2 Our contributions Our main contribution is to show that two of the main tools for the design of approximation schemes, namely approximate Pareto curves and Lagrangian relaxation, can lead to monotone and thus truthful algorithms.

Monotone construction of approximate Pareto curves. Our first contribution is a family of monotone multi-criteria FPTASs for the optimization problem $\mathcal{P}$ considered here, whose *exact version* admits a pseudo-polynomial-time algorithm $\mathcal{A}$ (see Section 3). We recall that, for a given weight function $\varphi : \mathcal{U} \to \mathbb{Q}^+$ and *target* B, the exact version of $\mathcal{P}$ consists of finding a feasible solution $S \in \mathcal{S}$ of weight $\varphi(S) = \sum_{e \in S} \varphi(e) = B$, if any. We implicitly assume that it is possible to remove all the solutions containing a given $e \in \mathcal{U}$ from $\mathcal{S}$ in polynomial time such that the resulting problem is of the same form as the original one (in this case we say that e is *discarded*). This is trivially true for all the applications

[1] Sometimes in the literature a weaker notion of monotonicity is considered, where only $\ell_0(e)$ can change. Typically it is much simpler to adapt known approximation algorithms to achieve that type of monotonicity. For example this holds for the primal-dual algorithm in [10]. Also optimal solutions to linear programs are monotone in this sense (see [1]). However, only the stronger type of monotonicity guarantees truthfulness with respect to all the $\ell_i(e)$'s.

considered in this paper. Our approximation schemes compute, for any given $\epsilon > 0$, a solution within a factor $(1+\epsilon)$ of the optimum which violates each budget constraint by a factor of at most $(1+\epsilon)$. The running time is polynomial in the size of the input and $1/\epsilon$.

THEOREM 1.1. *There is a truthful multi-criteria FPTAS for multi-objective problems $\mathcal{P}$ admitting a pseudo-polynomial-time algorithm for their exact version.*

Our result implies truthful FPTAS for a number of natural games. For example this covers the mentioned budgeted minimum spanning tree and budgeted shortest paths problems, and their generalizations with multiple budgets (possibly with budget lower bounds as well, which can be used to enforce, e.g., minimum quality standards). This generalizes the results in [5]. In the relevant special case of the *budgeted (perfect) matching* problem, where $\mathcal{S}$ is the set of matchings or perfect matchings of a graph G, the unique pseudo-polynomial-time exact algorithm known is Monte-Carlo [17] (a deterministic algorithm is known when G is planar [2]). In particular, with small probability this algorithm might fail to find an existing exact solution. In this case we can still apply our technique, but the resulting FPTAS is only probabilistically truthful. We recall that a randomized mechanism is *probabilistically truthful* if saying the truth is a dominant strategy with high probability (see, e.g., [1]). Using the reduction to matching in [2], we can achieve the same result for the *budgeted cycle packing* problem: find a node-disjoint packing of cycles of minimum cost and subject to budget constraints. The Monte-Carlo algorithm in [6] implies a probabilistically truthful FPTAS for the multi-budgeted version of the problem of finding a basis in the intersection of two matroids.

We remark that our approach also implies truthful multi-criteria FPTASs for multi-dimensional knapsack and for multi-unit combinatorial auctions for unknown multi-minded bidders with same valuation for each alternative and a fixed number of goods. This extends and improves on the results in [5] (see Section 1.1).

Our monotone FPTAS builds up on the construction of approximate Pareto curves by Papadimitriou and Yannakakis [19]. A Pareto solution S for a multi-objective problem is a solution such that there is no other solution S' which is as good as S on all objectives, and strictly better on at least one objective. The (potentially exponentially big) set of Pareto solutions defines the Pareto curve. [19] introduces the notion of ϵ-approximate Pareto-curve, i.e. a subset of solutions such that every Pareto solution is within a factor $(1+\epsilon)$ on each objective from some solution in the approximate curve. In the same paper the authors describe a general framework to construct such a curve in polynomial time in the size of the problem and $1/\epsilon$: a sufficient condition to do that is the existence of a pseudo-polynomial-time algorithm $\mathcal{A}$ for the *exact* version of the underlying feasibility problem. Their approach implies a multi-criteria FPTAS for the associated optimization problems where all the objectives but one are turned into budget constraints.

In this paper we show how to make monotone the FPTAS above. To that aim, we need to solve different subproblems, and extract the best solution obtained. Unfortunately, even if we use a monotone algorithm to solve each subproblem, the overall algorithm is not necessarily monotone. A similar issue is addressed in [5, 16], by defining a property strictly stronger than monotonicity: *bitonicity*. Roughly speaking, a monotone algorithm is bitonic if the cost of the solution *improves* when the set of parameters *improves* and vice versa. [5] shows that a proper combination of bitonic algorithms is monotone and we use the same basic approach.

Monotone Lagrangian relaxation. The approach above relies on the existence of a pseudo-polynomial-time exact algorithm and violates budget constraints by a $(1+\epsilon)$ factor. The second contribution of this paper is a technique based on the Lagrangian relaxation method which achieves (strict) budget feasibility but which cannot be used as a black-box as the FPTAS above (i.e., it requires the development of complementary, problem-specific, techniques). Using this alternative approach, we design a monotone randomized PTAS for the budgeted minimum spanning tree problem. This implies a universally truthful mechanism for the corresponding problem (see Section 4). We recall that a randomized mechanism is *universally truthful* if saying the truth is a dominant strategy for every outcome of the random bits. Universal truthfulness is the strongest form of truthfulness for randomized mechanisms [9, 18].

THEOREM 1.2. *There is a universally truthful Las Vegas PTAS for BMST.*

For a comparison, [5] gives a truthful FPTAS for the same problem on bounded-treewidth graphs, while the deterministic PTAS in [20] is not truthful.

The basic idea is combining our framework based on the composition of bitonic algorithms with the (non-monotone) PTAS of Goemans and Ravi [20]. The authors consider the Lagrangian relaxation of the problem with respect to the optimal Lagrangian multiplier (see, e.g. [13] and references therein for the Lagrangian relaxation method). Among the solutions of optimal Lagrangian cost, they return a feasible solution S^- which is adjacent to an infeasible solution S^+ (still of optimal

Lagrangian cost). The authors show that $\ell_0(S^-)$ exceeds the cost of the minimum spanning tree at most by the cost of one edge. In order to get a PTAS, they simply guess the $1/\epsilon$ most expensive edges of the optimal solution in a preliminary step, and reduce the problem consequently.[2]

Standard guessing is not compatible with monotonicity (and hence with bitonicity). To see that, consider the case that the modified element f is part of the guess, and hence the edges of cost larger than f (other than guessed edges) are removed from the graph. Then, decreasing the cost of f too much, leads to an infeasible problem where all non-guessed edges are discarded. In order to circumvent this problem, we developed a novel guessing step, where, besides guessing heavy edges, we also guess approximately their cost. This might seem counter-intuitive since one could simply consider their real cost, but it is crucial to prove bitonicity. We also remark that this has no (significant) impact on the running time and approximation factor of the algorithm, since the number of guesses remains polynomial and at least one of them correctly identifies the heaviest edges and their approximate cost. As far as we know, this is the first time that guessing is implemented in a monotone way: this might be of independent interest.

However, this is still not sufficient to achieve our goal. In fact, with Ravi and Goemans approach, several candidate solutions might be output. This introduces symmetries which have to be broken in order to guarantee bitonicity. One possibility is choosing, among the candidate solutions, the one with the largest cost. Also in this case, choosing the worst-cost solution might seem counter-intuitive. However, it is crucial to achieve bitonicity, and it has no impact on the (worst-case) approximation factor. The desired solution can be found by exploring exhaustively the space of the solutions of optimal Lagrangian cost. In fact, these solutions induce a connected graph which can be visited, starting from any such solution, in polynomial time in the size of the graph. The difficulty here is that this graph can be exponentially big. We solve this problem by means of randomization, in a way pretty close to the smoothed analysis of (standard) algorithms [23]. More specifically, we randomly perturb the input instance by multiplying each cost by a random, independent factor (close to one, to preserve the approximation ratio). After this perturbation, with high probability there will be only two optimal Lagrangian solutions. As a consequence, the running time of the algorithm is polynomial in expectation (the algorithm is Las-Vegas). For our mechanism viewed as a probability distribution over deterministic mechanisms (one for each possible value of the perturbation) to be universally truthful we have to prove that each of these deterministic algorithms is monotone (in fact bitonic). We prove this by a kind of sensitivity analysis of 2-dimensional packing linear programs that correspond to the lower envelope of lines representing spanning trees in the Lagrangian relaxation approach of [20]. This analysis differs from the standard packing LP sensitivity analysis (see, e.g., [22]) in that we also change an entry of the LP matrix and not only the right-hand-side constants.

We can combine the monotone guessing step above with a slight variant[3] of the truthful FPTAS described before to obtain truthful PTASs for problems which are *inclusion-closed*, i.e. such that removing any element from a feasible solution preserves feasibility. These PTASs assume that we can remove in polynomial time all the solutions in $\mathcal{S}$ not containing a given element $e \in \mathcal{U}$. An application of inclusion-closed problems satisfying this assumption is, for example, the problem of computing a (possibly non-perfect) matching of maximum cost, subject to budget upper bounds. The proof of the following theorem will be given in the full version of the paper.

THEOREM 1.3. *There is a (probabilistically) truthful PTAS for inclusion-closed multi-objective problems $\mathcal{P}$ admitting a pseudo-polynomial-time (Monte-Carlo) algorithm $\mathcal{A}$ for their exact version.*

2 Preliminaries and notation

As noted above, in this work we consider the case in which the type $\ell^f = (\ell_0(f), \ell_1(f), \ldots, \ell_k(f))$ of each agent f is private information. We consider the general framework of *generalized single-minded* agents introduced in [5]. For a given algorithm A, let S' be the solution given in output by A on input declarations $(\ell'^f)_{f\in\mathcal{U}}$. Agent f with type ℓ^f evaluates solution S' according to the valuation function described in Figure 1. A mechanism augments algorithm A with a payment function p. The mechanism (A, p) on input $(\ell'^f)_{f\in\mathcal{U}}$ computes solution S' and awards agent f a payment $p^f((\ell'^f)_{f\in\mathcal{U}})$ which is non-positive if $best = \max$ and non-negative when $best = \min$. Agent f derives utility

$$u^f((\ell'^f)_{f\in\mathcal{U}}, \ell^f) = v^f(S', \ell^f) + p^f((\ell'^f)_{f\in\mathcal{U}}).$$

[2] The PTAS in [20] is presented to return an infeasible solution adjacent to a feasible one. The guessing is then done on the longest edges of the optimal solution. The two approaches are actually equivalent approximation-wise but not for monotonicity. (It is easy to provide a counterexample for the original one.)

[3] A similar approach is used to obtain a non-monotone randomized PTAS for the multi-budgeted matching problem in [11].

$$v^f(S', \ell^f) = \begin{cases} \ell_0(f) sign_0 & \text{if } f \in S' \wedge_{i=1}^{k} \ell'_i(f) \preceq_i \ell_i(f), \\ -\infty & \text{if } f \in S' \wedge \ell'_i(f) \not\preceq_i \ell_i(f) \text{ for some } i \in \{1, \ldots, k\}, \\ 0 & \text{otherwise.} \end{cases}$$

Figure 1: Valuation function of generalized single-minded agents.

The mechanism is truthful if, for each agent f, the utility is maximized by truthtelling, i.e.,

$$u^f((\ell^f, \ell'^{-f}), \ell^f) \geq u_f((\ell'^f)_{f \in \mathcal{U}}, \ell^f)$$

for each ℓ'^f and ℓ'^{-f} with ℓ'^{-f} denoting the declarations of all agents but f. [5, 15] show that designing a *monotone* algorithm is sufficient to obtain a truthful mechanism for agents with valuation functions as above. We recall that algorithm A is monotone if $f \in S'$ implies that $f \in \bar{S}'$ with $\bar{S}'$ denoting the solution computed by A on input $(\bar{\ell}'^f, \ell'^{-f})$ for any $\bar{\ell}'^f$ such that $\bar{\ell}'_i(f) \succeq_i \ell'_i(f)$ for some $i \in \{0, 1, \ldots, k\}$.

PROPOSITION 2.1. ([5, 15]) *For generalized single-minded agents, a monotone algorithm A admits a payment function p such that (A, p) is a truthful mechanism for $\mathcal{P}$.*[4]

Generalized single-minded agents adapt well to our setting. Consider again the case in which $\mathcal{P}$ models BMST problem and $\ell_0(f)$ and $\ell_1(f)$ are respectively the cost and the delay experienced when using edge f. As noted, agent f cannot lie promising a delay smaller than her true one (or otherwise the mechanism can *a posteriori* verify that the agent lied). In the valuation function above, this is reflected by a valuation of $-\infty$ for declarations in which agents underbid the delay. (In other words, an agent would incur an infinite cost to provide the service with a smaller delay.) More generally, the valuation functions that we consider model optimization problems $\mathcal{P}$ in which budgeted parameters are somehow verifiable (thus implying that certain lies are irrational for a selfish agent).

Valuation function in Figure 1 connects with two different research areas in Algorithmic Mechanism Design. Mechanisms with verification, introduced by [18], exploit the observation that the execution of the mechanism can be used to verify agents misreporting their types (i.e., the entire type must be verifiable). On the contrary, in our framework this assumption is only made for budgeted parameters, i.e., $\ell_0(\cdot)$'s can model unverifiable quantities (e.g., costs). Valuation function of Figure 1 expresses the valuation of single-minded bidders in a Combinatorial Auction (considered the paradigmatic problem in Algorithmic Mechanism Design) using exact[5] allocation algorithms. Indeed, it is enough to consider the budgeted parameter as the demand set (i.e., a bidder evaluates $-\infty$ a set which is not a superset of her unique demand set).

We conclude that generalized single-minded agents is a general framework motivated by Combinatorial Auctions that well encompasses the multi-objective optimization problems we consider. However, observe that if we would allow unrestricted ways of lying on budgeted parameters then even a VCG mechanism (using an *exponential-time* optimal algorithm) would not be truthful for a multi-objective optimization problem $\mathcal{P}$. To see this, consider again the BMST problem and take as instance a triangle graph with delay budget L. Name the three edges of the graph e_1, e_2 and e_3 and assume the true types are $\ell^{e_1} = (\epsilon, \epsilon)$, $\ell^{e_2} = (0, 0)$ and $\ell^{e_3} = (0, H)$ for some $0 < \epsilon \leq L$ and $H > L$. When agents are truthtelling the VCG mechanism would select the only feasible tree comprised of edges e_1 and e_2. The utility of edge e_3 is in this case 0. Now consider edge e_3 misreporting her type as follows $\ell'^{e_3} = (0, L)$. In this case the VCG mechanism would select the tree comprised by edges e_2 and e_3 (i.e., the minimum cost tree among the seemingly feasible ones) and pay agent e_3 an amount of $\epsilon > 0$. Since the true cost of e_3 is 0 then e_3 has a strict improvement of her utility by lying. (Note that on the contrary in our generalized single-minded setting the valuation of e_3 would be $-\infty$ in this case.) VCG fails since there is a part of agents' type (namely, the delay) whose lies are not reflected in the objective function considered by VCG (which is only the sum of the costs). Note, however, that overbidding the delay can only shrink the set of seemingly feasible solutions and thus gives no advantages to unselected agents. Indeed, as observed above VCG is truthful (though not efficient) in our generalized single-minded setting. Finally, we remark that the example described above can be tweaked to show that no algorithm with polynomially bounded approximation guarantee can be used to obtain a truthful mechanism satisfying a strict form of *voluntary participation* (i.e., a mechanism in which se-

[4] The results in [5, 15] also require the algorithm to be *exact*. However, in our context any algorithm for $\mathcal{P}$ is exact due to the constraint $x_e \in \{0, 1\}$.

[5] In CA's terminology, an allocation algorithm is exact if it allocates to each bidder the declared set or none.

lected agents always have a strictly positive utility). It is enough to consider the same instance with $\ell^{e_1} = (M, \epsilon)$ and $\ell^{e_2} = (1, 0)$. Any M-approximation algorithm in input ℓ^{e_1}, ℓ^{e_2} and ℓ'^{e_3} must select the tree comprised of edges e_2 and e_3.

Notation. For notational convenience, we assume that our algorithms, in case they are not able to find a solution, anyway return a special null solution $\mathcal{N}$, whose cost is $-\infty$ for *best* = max, and $+\infty$ otherwise. For ease of presentation, we will assume that costs, lengths, and budgets are strictly positive, and the optimum solution OPT is not empty. These assumptions can be removed at the cost of more technical algorithms and analysis. We will denote by f the element whose cost/length is modified as by monotonicity condition, and by e a generic element. We use an upper bar to denote quantities in the modified problem.

Composition of bitonic algorithms. Consider a cost function $c : 2^{\mathcal{U}} \to \mathbb{Q}^+$, and let $\succeq$ be a corresponding order relation. In the following we will always assume $\succeq = \succeq_0$, while the choice of $c(\cdot)$ depends on the context. For a given algorithm A, let $\mathcal{P}$ be the original problem. By $\bar{\mathcal{P}}$ we denote a problem in which f declares $\bar{\ell}_i(f) \succeq_i \ell(f)$ for some $i = 0, \ldots, k$. We let S and $\bar{S}$ to be the solutions returned by A on problems $\mathcal{P}$ and $\bar{\mathcal{P}}$, respectively.

DEFINITION 2.1. *Algorithm A is* bitonic *with respect to $c(\cdot)$ if the following properties hold:*

(i) If $f \in S$, then $f \in \bar{S}$ and $\bar{c}(\bar{S}) \succeq c(S)$.

(ii) If $f \notin S$, then either $f \in \bar{S}$ or $\bar{c}(\bar{S}) \preceq c(S)$.

Observe that a bitonic algorithm is monotone, while the vice versa is not necessarily true.

As shown in [5], bitonic algorithms can be combined to get a monotone algorithm.

THEOREM 2.1. (COMPOSITION THEOREM) [5] *Consider a procedure $\mathcal{M}$ which generates a (potentially infinite) family of subproblems $P_1, P_2, \ldots$, solves each subproblem P_i with a procedure $\mathcal{M}_i$ and returns the solution S_i to problem P_i minimizing (resp., maximizing) $c_i(S_i)$, for given cost functions $c_i(\cdot)$. (The best solution with largest index i in case of ties). If each procedure $\mathcal{M}_i$ is bitonic with respect to $c_i(\cdot)$, then $\mathcal{M}$ is monotone.*

3 Approximate Pareto curves and monotone multi-criteria FPTASs

In this section we restrict our attention to multi-objective optimization problems $\mathcal{P}$ whose exact version admits a pseudo-polynomial-time algorithm $\mathcal{A}$. For these problems we describe a monotone multi-criteria FPTAS `multi`. Our approach is inspired, approximation-wise, by the construction of ϵ-approximate Pareto curves in [19], and crucially exploits the combination of bitonic procedures to achieve monotonicity.

3.1 Algorithm Algorithm `multi` is described in Figure 2. The basic idea is generating a family of subproblems $\mathcal{P}_1, \ldots, \mathcal{P}_q$. Each subproblem $\mathcal{P}_j$ is solved by means of a procedure `feasible`, hence obtaining a solution S_j and a cost function $\ell_{0,j}(\cdot)$ (Step M1). This procedure is designed in order to be bitonic with respect to $\ell_{0,j}(\cdot)$ on subproblem $\mathcal{P}_j$. Eventually `multi` returns the solution S_h optimizing $\ell_{0,h}(S_h)$, the *best* solution with largest index h in case of ties (Step M2).

In more detail, let $\ell_{0,min}$ and $\ell_{0,max}$ be the smallest and largest cost ℓ_0, respectively. We define $B_{0,1} \preceq_0 \ldots \preceq_0 B_{0,q}$ to be all the (positive and/or negative) powers of $(1+\epsilon)$ between proper boundary values. For each $B_{0,j}$, $\mathcal{P}_j$ is the feasibility problem obtained by considering the set of constraints of the original problem $\mathcal{P}$, plus the constraint $\sum_{e \in \mathcal{U}} \ell_0(e) x_e \succeq B_{0,j}$.

For a given subproblem $\mathcal{P}_j$, procedure `feasible` assumes $B_0 := B_{0,j}$. The aim of the procedure is computing an ϵ-feasible solution to $\mathcal{P}_j$, i.e. a solution $S_j \in \mathcal{S}$ such that, for all $i \geq 0$, $\ell_i(S_j) \succeq_i B_i/(1+\epsilon)^{sign_i}$. Moreover, `feasible` returns a cost function $\ell_{0,j}(\cdot)$ such that the behavior of `feasible` is bitonic with respect to function $\ell_{0,j}(\cdot)$ on subproblem $\mathcal{P}_j$ (more details in the proof of Lemma 3.2). In order to achieve this goal, `feasible` first constructs an auxiliary feasibility problem $\mathcal{P}'_j$, with lengths $\ell'_i(e)$ and budgets B'_i which are polynomially-bounded in m/ϵ (Steps F1 and F2). The definitions of those auxiliary lengths and budgets are such that any feasible (w.r.t. $\ell_i(\cdot)$, $i \geq 1$) solution for $\mathcal{P}_j$ is feasible for $\mathcal{P}'_j$, and every feasible solution to $\mathcal{P}'_j$ is ϵ-feasible (w.r.t. $\ell_i(\cdot)$, $i \geq 1$) for $\mathcal{P}_j$ (more details in the proof of Lemma 3.1). Then `feasible` finds a feasible solution for $\mathcal{P}'_j$ by encoding this problem in a proper family of exact problems, which are solved by means of $\mathcal{A}$ (Step F3). Each exact problem is indexed by a vector $z = (z_0, \ldots, z_k)$ in a proper domain $\times_{i=0}^{k} I_i$. Consider the *length vector* $\Lambda(\cdot) = (\ell'_0(\cdot), \ldots, \ell'_k(\cdot))$. The solution S_z returned for the exact problem indexed by z, if non-null, satisfies $\Lambda(S_z) = z$. The domain $\times_{i=0}^{k} I_i$ is defined by the set of vectors z such that $z_i \succeq_i B'_i$ for all i. Hence, if there is a feasible solution to $\mathcal{P}'_j$, i.e. a solution S_j with $\ell'_i(S_j) \succeq_i B'_i$ for all i, this solution will be found by `feasible`. Eventually (Step F4), among the feasible solutions S_z obtained, `feasible` returns the solution optimizing in lexicographic sense z, with the lexicographic order $\succeq$ induced by the $\succeq_i$'s. Choosing the

`multi`$(\mathcal{P}, \epsilon)$

M1: Let $B_{0,1} \preceq \ldots \preceq B_{0,q}$ be the powers of $(1+\epsilon)$ between $\ell_{0,min}\frac{1}{m(1+\epsilon)}$ and $m\,\ell_{0,max}\lceil\frac{m(1+\epsilon)}{\epsilon}\rceil$. For $j = 1, \ldots, q$, let $(S_j, \ell_{0,j}(\cdot)) =$ `feasible`$(\mathcal{P}_j, \epsilon, B_{0,j})$.

M2: Return the solution $S^* = S_h$ optimizing $\ell_{0,h}(S_h)$, the *best* solution with largest index h in case of ties.

`feasible`$(\mathcal{P}, \epsilon, B_0)$

F1: For all $e \in \mathcal{U}$:

F1a: If $\succeq_0 = \geq$: $B'_0 = \lceil\frac{m}{\epsilon}\rceil$, $I_0 = \{B'_0, B'_0+1, \ldots, m\,B'_0\}$. $\ell_0(e) := \min\{\ell_0(e), B_0\}$. Let $\ell'_0(e) = \lfloor\frac{B'_0}{B_0}\ell_0(e)\rfloor$.

F1b: If $\succeq_0 = \leq$: $B'_0 = \lceil\frac{m(1+\epsilon)}{\epsilon}\rceil$, $I_0 = \{0, 1, \ldots, B'_0\}$. Discard all $\{e : \ell_0(e) > B_0\}$. Let $\ell'_0(e) = \lceil\frac{B'_0}{B_0}\ell_0(e)\rceil$.

F2: For $i = 1, \ldots, k$, for all $e \in \mathcal{U}$:

F2a: If $\succeq_i = \geq$: $B'_i = \lceil\frac{m(1+\epsilon)}{\epsilon}\rceil$, $I_i = \{B'_i, B'_i+1, \ldots, m\,B'_i\}$. $\ell_i(e) := \min\{\ell_i(e), B_i\}$. Let $\ell'_i(e) = \lceil\frac{B'_i}{B_i}\ell_i(e)\rceil$.

F2b: If $\succeq_i = \leq$: $B'_i = \lceil\frac{m}{\epsilon}\rceil$, $I_i = \{0, 1, \ldots, B'_i\}$. Discard all $\{e : \ell_i(e) > B_i\}$. Let $\ell'_i(e) = \lfloor\frac{B'_i}{B_i}\ell_i(e)\rfloor$.

F3: Let $M = m\,\max_{i\geq 0}\{B'_i\} + 1$ and $\varphi(\cdot) = \sum_{i=0}^{k} M^i \ell'_i(\cdot)$. For all $z = (z_0, z_1, \ldots, z_k) \in \times_{i=0}^{k} I_i$:

- Let $B_z = \sum_{i=0}^{k} M^i z_i$. Use $\mathcal{A}$ to compute a solution $S_z \in \mathcal{S}$ with $\varphi(S_z) = B_z$. (The solution with lexicographically largest incidence vector in case of ties.) If no feasible solution exists, $S_z = \mathcal{N}$.

F4: Let $S^*_z \neq \mathcal{N}$ be the solution S_z with lexicographically *best* z ($S^*_z = \mathcal{N}$ if no such solution exists). Return $(S^*_z, \frac{B_0}{B'_0}\ell'_0(\cdot))$.

Figure 2: Algorithm `multi`.

best z in lexicographic sense, besides providing a good approximation ratio (due to the fact that we optimize z_0 first), is crucial to enforce bitonicity.

Note that in Step F3, in case of multiple solutions of target value B, the algorithm returns the solution of largest incidence vector. This tie breaking rule is also crucial to achieve bitonicity. It is easy to modify a given exact algorithm $\mathcal{A}$ to enforce this property. Let $e_1, e_2, \ldots, e_m$ be the elements, and $B' = B$. For $i = 1, \ldots, m$, we add a very large (but still polynomial) value L to $\varphi(e_i)$, and ask for a solution of target value $L + B'$. For L large enough, any such solution must contain e_i. If no such solution exists, we discard e_i. Otherwise, we set $B' \leftarrow B' + L$. In both cases we proceed with next edge. Here we exploit the assumption that discarding one element does not change the nature of the considered problem.

3.2 Analysis Let us bound the running time and approximation factor of `multi`.

LEMMA 3.1. *Algorithm* `multi` *computes a* $(1+\epsilon)^2$*-approximate solution, violating each budget constraint at most by a factor* $(1+\epsilon)$*. The running time of the algorithm is polynomial in the size of the input and* $1/\epsilon$*.*

Proof. Consider the running time of `feasible` on a given subproblem $\mathcal{P}_j$. Lengths $\ell'_i(\cdot)$ and budgets B'_i are polynomially bounded in m/ϵ. Consequently, the number of exact problems generated to solve $\mathcal{P}'_j$ is $O((m/\epsilon)^k) = O((m/\epsilon)^{O(1)})$ since k is constant. Also, the fact that $k = O(1)$ implies that the values of $f(\cdot)$ and B of each exact problem satisfy the same bound. Since $\mathcal{A}$ is a pseudo-polynomial-time algorithm, it follows that each exact problem can be solved in $O((m/\epsilon)^{O(1)})$ time. Altogether, the running time of `feasible` is $O((m/\epsilon)^{O(1)})$. The number of subproblems generated by `multi` is $O(\log_{1+\epsilon}\frac{m\,\ell_{0,max}}{\epsilon\ell_{0,min}})$, which is polynomial in the size of the input and $1/\epsilon$. The running time bound follows.

Consider now the approximation factor. We initially observe that, for any solution S returned for some subproblem $\mathcal{P}'_j$, and for any $i \geq 1$,

$$\ell_i(S) = \frac{B_i}{B'_i}\sum_{e\in S}\frac{B'_i}{B_i}\ell_i(e) \succeq_i \frac{B_i}{B'_i}\sum_{e\in S}(\ell'_i(e) - sign_i)$$
$$\succeq_i B_i - B_i\frac{sign_i}{B'_i}m \succeq_i \frac{B_i}{(1+\epsilon)^{sign_i}}.$$

In particular, the solution S^* returned by the algorithm violates each budget constraint at most by a factor $(1+\epsilon)$. We next show that $S_h := S^*$ is $(1+\epsilon)^2$-approximate.

Let OPT be an optimum solution to the original problem instance and let j be the largest index such that $\ell_0(OPT) \succeq_0 B_{0,j}(1+\epsilon)^{sign_0}$. In particular, $\ell_0(OPT) \prec_0 B_{0,j+1}(1+\epsilon)^{sign_0} = B_{0,j}(1+\epsilon)^{2sign_0}$.

Observe that there is one such value $B_{0,j}$, since $\ell_{0,min} \leq \ell_0(OPT) \leq m\,\ell_{0,max}$. Note that OPT is feasible for $\mathcal{P}'_j$, i.e., $\ell'_i(OPT) \succeq_i B'_i$ for $i \geq 0$. To see this consider an execution of $(S_j, \ell_{0,j}(\cdot)) = \texttt{feasible}(\mathcal{P}_j, \epsilon, B_{0,j})$ in algorithm `multi`. Observe that $\ell'_0(\cdot) = \ell_{0,j}(\cdot)\frac{B'_{0,j}}{B_{0,j}}$, $B_0 = B_{0,j}$, and $B'_0 = B'_{0,j}$. For any $i \geq 1$,

$$\ell'_i(OPT) = \sum_{e \in OPT} \ell'_i(e) \succeq_i \sum_{e \in OPT} \frac{B'_i}{B_i}\ell_i(e) \succeq_i \frac{B'_i}{B_i}B_i = B'_i.$$

Moreover,

$$\ell'_0(OPT) = \sum_{e \in OPT} \ell'_0(e) \succeq_0 \sum_{e \in OPT} \frac{B'_0}{B_0}\ell_0(e) - sign_0 \succeq_0 B'_0(1+\epsilon)^{sign_0} - sign_0\, m \succeq_0 B'_0.$$

This implies that a solution S_j is returned for subproblem $\mathcal{P}_j$. By the optimality of S_h, $\ell_{0,h}(S_h) \succeq_0 \ell_{0,j}(S_j)$. It follows that

$$\frac{B_{0,h}}{B'_{0,h}}\ell'_{0,h}(S_h) = \ell_{0,h}(S_h) \succeq_0 \ell_{0,j}(S_j) = \frac{B_{0,j}}{B'_{0,j}}\ell'_{0,j}(S_j) \succeq_0 B_{0,j} \succeq_0 \frac{\ell_0(OPT)}{(1+\epsilon)^{2sign_0}}.$$

The cost of $S^* = S_h$ then satisfies

$$\ell_0(S_h) = \frac{B_{0,h}}{B'_{0,h}} \sum_{e \in S_h} \frac{B'_{0,h}}{B_{0,h}}\ell_0(e) \succeq_0 \frac{B_{0,h}}{B'_{0,h}} \sum_{e \in S_h} \ell'_{0,h}(e) = \frac{B_{0,h}}{B'_{0,h}}\ell'_{0,h}(S_h) \succeq_0 \frac{\ell_0(OPT)}{(1+\epsilon)^{2sign_0}}.$$

□

We next prove the bitonicity of `feasible`: via the Composition Theorem 2.1 this will imply the monotonicity of `multi`.

LEMMA 3.2. *Procedure* `feasible` *is bitonic with respect to* $\ell_{0,j}(\cdot)$ *on subproblem* $\mathcal{P}_j$.

Proof. To fix notation, consider an execution of $(S_j, \ell_{0,j}(\cdot)) = \texttt{feasible}(\mathcal{P}_j, \epsilon, B_{0,j})$ in algorithm `multi`. Observe that $\ell'_0(\cdot) = \ell_{0,j}(\cdot)\frac{B'_{0,j}}{B_{0,j}}$, $B_0 = B_{0,j}$, and $B'_0 = B'_{0,j}$. Let also $\bar{S}_j$ be the solution output by `feasible` for problem $\bar{\mathcal{P}}_j$.

Since $\ell_{0,j}(\cdot) = \frac{B_0}{B'_0}\ell'_0(\cdot)$, it is sufficient to prove bitonicity with respect to $\ell'_0(\cdot)$. Suppose we modify $\ell_s(f)$ to $\bar{\ell}_s(f) \succeq_s \ell_s(f)$ for some $s \in \{0, \ldots, k\}$. Observe that this implies $\bar{\ell}'_s(f) \succeq_s \ell'_s(f)$. Consider the case $\bar{\ell}'_s(f) = \ell'_s(f)$: here $S_j = \bar{S}_j$ and so the algorithm is trivially bitonic. Then assume $\bar{\ell}'_s(f) \succ_s \ell'_s(f)$. By $\mathcal{F}$ we denote the set of feasible solutions to $\mathcal{P}'_j$ computed by $\mathcal{A}$. Note that $\mathcal{F} \subseteq \bar{\mathcal{F}}$, since every feasible solution to $\mathcal{P}'_j$ is feasible for $\bar{\mathcal{P}}'_j$ as well. Moreover, every solution in $\bar{\mathcal{F}} \setminus \mathcal{F}$ must contain f.

Consider first the case $f \in S_j$. We will show that $f \in \bar{S}_j$ and $\bar{\Lambda}(\bar{S}_j) \succeq \Lambda(S_j)$ (which implies $\bar{\ell}'_0(\bar{S}_j) \succeq_0 \ell'_0(S_j)$). Note that $\bar{\Lambda}(S_j) \succ \Lambda(S_j)$ since $\bar{\ell}'_s(f) \succ_s \ell'_s(f)$. Moreover, because $S_j \in \mathcal{F} \subseteq \bar{\mathcal{F}}$ and the algorithm returns in Step F4 lexicographically best solution we have $\bar{\Lambda}(\bar{S}_j) \succeq \bar{\Lambda}(S_j)$. As a consequence, $\bar{\Lambda}(\bar{S}_j) \succeq \bar{\Lambda}(S_j) \succ \Lambda(S_j)$. On the other hand, for any $f \notin S' \in \mathcal{F}$, $\bar{\Lambda}(S') = \Lambda(S') \preceq \Lambda(S_j)$ (where $\preceq$ again follows from Step F4), and hence $S' \neq \bar{S}_j$. We can conclude that $f \in \bar{S}_j$.

Suppose now $f \notin S_j$ and $f \notin \bar{S}_j$. We will show that $\bar{S}_j = S_j$ (which implies $\bar{\ell}'_0(\bar{S}_j) \preceq_0 \ell'_0(S_j)$). Since all the solutions in $\bar{\mathcal{F}} \setminus \mathcal{F}$ contain f, $\bar{S}_j \in \mathcal{F}$. Then $\bar{\Lambda}(\bar{S}_j) = \Lambda(\bar{S}_j) \preceq \Lambda(S_j) = \bar{\Lambda}(S_j) \preceq \bar{\Lambda}(\bar{S}_j)$ (where we referred to Step F4 twice), which implies $\bar{\Lambda}(\bar{S}_j) = \Lambda(S_j)$. Therefore the set of solutions not containing f which optimize the length vector Λ is exactly the same in the two problems. This implies that $\bar{S}_j = S_j$ by the lexicographic optimality of the solutions computed. □

LEMMA 3.3. *Algorithm* `multi` *is monotone.*

Proof. Consider the variant `ideal` of `multi` which spans all the (infinitely many) powers of $(1+\epsilon)$. Observe that `ideal` and `multi` output exactly the same solution (both in the original and in the modified problem). In fact, consider the case $\succeq_0 = \geq$. For $B_0 < \frac{\ell_{0,min}}{m}$, we have $\ell'_0(e) = B'_0$ for all e, which leads to a solution of $\ell_{0,j}$ cost at most $\frac{B_0}{B'_0} m B'_0 < \ell_{0,min}$. On the other hand, for $B_0 > m\ell_{0,max}\lceil\frac{m(1+\epsilon)}{\epsilon}\rceil$, we have $\ell'_0(e) = 0$ for all e, which leads to a solution of $\ell_{0,j}$ cost zero. Observe that in both cases the solutions computed by `ideal` only are not better than the solutions computed by `multi`. The case $\succeq_0 = \leq$ is symmetric. For $B_0 < \ell_{0,min}$ all the edges are discarded and the problem becomes infeasible. For $B_0 > m\ell_{0,max}\lceil\frac{m(1+\epsilon)}{\epsilon}\rceil$, we have $\ell'_0(e) = 1$ for all e, which leads to a solution of $\ell_{0,j}$ cost at least $\frac{B_0}{B'_0} > m\ell_{0,max}$. Also in this case the solutions computed by `ideal` only are not better than the solutions computed also by `multi`. By Lemma 3.2 and the Composition Theorem 2.1, `ideal` is monotone. We can conclude that `multi` is monotone as well. □

Theorem 1.1 follows from Lemmas 3.1 and 3.3, by applying the results in [5, 15].

4 Lagrangian relaxation and budgeted minimum spanning tree

In this section we investigate a different approach to the design of truthful mechanisms, based on the classical Lagrangian relaxation method. With this approach, we obtain a monotone randomized PTAS `bmst` for the budgeted minimum spanning tree problem (BMST). This implies a universally truthful mechanism for the corresponding game.

Let us start by introducing some preliminary notions. For notational convenience, let $c(\cdot) = \ell_0(\cdot)$, $\ell(\cdot) = \ell_1(\cdot)$ and $L = B_1$. By c_{min} and c_{max} we denote the smallest and largest cost, respectively. BMST can be defined as follows:

$$\begin{aligned} \min \quad & \sum_{e\in E} c(e)x_e \\ s.t. \quad & x \in \mathcal{X} \\ & \sum_{e\in E} \ell(e)x_e \le L \end{aligned}$$

Here $\mathcal{X}$ denotes the set of incidence vectors of spanning trees of the input graph $G = (V, E)$. For a Lagrangian multiplier $\lambda \ge 0$, the Lagrangian relaxation of the problem is (see [20])

$$\begin{aligned} LAG(\lambda) = \min \quad & \sum_{e\in E} c(e)x_e + \lambda \cdot (\sum_{e\in E} \ell(e)x_e - L) \\ s.t. \quad & x \in \mathcal{X} \end{aligned}$$

The problem above is essentially a standard minimum spanning tree problem, with respect to the *Lagrangian costs* $c'(e) = c(e) + \lambda\,\ell(e)$. Observe that, for any $\lambda \ge 0$, $LAG(\lambda) \le OPT$. We let λ^* (*optimal Lagrangian multiplier*) be a value of $\lambda \ge 0$ which maximizes $LAG(\lambda)$. In case of a tie, we let λ^* be the smallest such value. If $\lambda^* = \infty$, there is no feasible solution. We remark that λ^* can be computed in strongly-polynomial-time $O(m^{O(1)})$, using, say, Megiddo's parametric search technique (see, e.g., [20]).

Function $LAG(\lambda)$ has a natural geometric interpretation. For any spanning tree S with incidence vector $x(S)$, $c_\lambda(S) := \sum_{e\in E} c(e)x_e(S) + \lambda \cdot (\sum_{e\in E} \ell(e)x_e(S) - L)$ is a linear function of the multiplier λ. In particular, the slope of $c_\lambda(S)$ is positive if S is infeasible, and non-positive otherwise. $LAG(\lambda)$ is the lower envelope of the lines $c_\lambda(S)$, for $\lambda \ge 0$ and $S \in \mathcal{S}$. Observe that $LAG(\lambda)$ is concave and piecewise linear. When no confusion is possible, we use the notion of spanning tree and line interchangeably. We observe the following useful fact.

LEMMA 4.1. *Consider the lower envelope LE of a set of solutions. Let $S^1, S^2, \ldots, S^q$ be the solutions intersecting LE, sorted in decreasing order of length (breaking ties arbitrarily). Then, for $i < j$, $c(S^i) \le c(S^j)$.*

Proof. It is sufficient to show that, for any $i = 1, 2, \ldots, q-1$, $c(S^i) \le c(S^{i+1})$. If $S^i(\lambda)$ and $S^{i+1}(\lambda)$ overlap, the claim is trivially true. Otherwise, let λ' be the value of λ for which $S^i(\lambda') = S^{i+1}(\lambda')$ (the two lines cannot be parallel, since they both intersect LE). Recall that $\ell(S^i) \ge \ell(S^{i+1})$ by assumption. Then

$$\begin{aligned} c(S^i) =& c_{\lambda'}(S^i) - \lambda'(\ell(S^i) - L) \\ \le& c_{\lambda'}(S^i) - \lambda'(\ell(S^{i+1}) - L) \\ \le& c_{\lambda'}(S^{i+1}) - \lambda'(\ell(S^{i+1}) - L) = c(S^{i+1}). \end{aligned}$$

□

4.1 Algorithm Algorithm `bmst` is described in Figure 3. The first step of the construction is a random perturbation of the costs (Step B1). The factor $T = 2^m$ in the perturbation is simply an upper bound on the number of spanning trees. Our perturbation will ensure, with high probability, that no more than 2 lines corresponding to spanning trees intersect at a given point.

Then (Step B2), the algorithm generates a polynomial set of subproblems $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_h$. Intuitively, each subproblem corresponds to a guess of the most expensive edges in the optimum solution OPT, and to a guess of their approximate cost. More precisely, each subproblem j is labeled by a pair $(F_j, g_j(\cdot))$, where F_j is a subset of $1/\epsilon$ edges, and $g_j : F_j \to C$, where $C = \{c_1, \ldots, c_q\}$ is a proper set of rounded costs. Given a pair $j = (F_j, g_j(\cdot))$, subproblem $\mathcal{P}_j$ is obtained by removing F_j and all the edges of cost larger than $\min_{e\in F_j}\{g_j(e)\}$, and decreasing L by $\ell(F)$.

Each subproblem $\mathcal{P}_j$ is solved by means of a procedure `lagrangian`, based on the Lagrangian relaxation method. In particular, following [20], among the solutions intersecting LAG at λ^*, we select two adjacent solutions S^- and S^+, of non-positive and positive slope respectively, and return $S_j = S^-$. We recall that two spanning trees S^- and S^+ are adjacent in the spanning tree polytope if there are edges e^+ and e^- such that $S^- = S^+ \setminus \{e^+\} \cup \{e^-\}$. As mentioned in the introduction, in case of ties we select the pair (S^-, S^+) with maximum $c(S^-)$.

Eventually (Step B3), `bmst` returns the (feasible) solution $F_j \cup S_j$ of minimum cost $c(F_j \cup S_j) = c(F_j) + c(S_j)$ (and largest index j in case of ties).

We remark that the values t_e in the perturbation are independent from edge costs and lengths, and hence we can assume that they are the same in the original ($\mathcal{P}_j$) and modified ($\bar{\mathcal{P}}_j$) problems. In other words, each choice of the t_e's specifies one deterministic algorithm. We will show that each such algorithm is monotone. Note also that monotonicity on the perturbed instance implies monotonicity on the original instance, since the

`bmst`$(\mathcal{P}, \epsilon)$

B1: For all $e \in E$, independently generate an integer t_e uniformly at random in $\{1, \ldots, 2T^4\}$, $T = 2^m$, and replace $c(e)$ with $c(e)(1 + \epsilon \frac{t_e}{2T^4})$

B2: Let $c_1 \leq \ldots \leq c_q$ be the powers of $(1+\epsilon)$ between $c_{min}/(1+\epsilon)$ and $c_{max}(1+\epsilon)$. Let $1, \ldots, h$ denote all the pairs $(F, g(\cdot))$ with $F \subseteq E$, $|F| = \frac{1}{\epsilon}$, and $g : F \to \{c_1, \ldots, c_q\}$. For a given pair $j = (F_j, g_j(\cdot))$, define subproblem $\mathcal{P}_j$ by removing from G, F_j and all the edges of value larger than $\min_{e \in F_j}\{g_j(e)\}$ and decreasing B by $\ell(F_j)$. Compute $S_j =$ `lagrangian`$(\mathcal{P}_j)$.

B3: Return the solution $F_j \cup S_j$ minimizing $c(F_j) + c(S_j)$, and maximizing j in case of ties.

`lagrangian`$(\mathcal{P}_j)$

L1: Compute the optimal Lagrangian multiplier λ^*, the smallest one in case of ties.

L2: If $\lambda^* = 0$, return the solution S^- of minimum-slope intersecting LAG at λ^*.

L3: If $\lambda^* = +\infty$, return $\mathcal{N}$.

L4: Compute a pair of adjacent solutions S^- and S^+, of non-positive and positive slope, respectively. In case of ties, the pair where $c(S^-)$ is maximized and, as a second choice, the incidence vector of S^- is minimized. Return S^-.

Figure 3: Algorithm `bmst`.

perturbation does not change the sign of the difference between modified and original costs/lengths. As a consequence, the resulting mechanism will be universally truthful. The random perturbation has the following property.

LEMMA 4.2. *After the perturbation, with probability at least $1 - 1/T$, all the spanning trees correspond to different lines, and at most two lines intersect at a given point.*

Proof. Consider any two lines $S_1(\lambda)$ and $S_2(\lambda)$, and let ℓ_i and c_i be the length and perturbed cost of S_i, respectively. The two lines overlap if and only if $c_1 = c_2$ and $\ell_1 = \ell_2$. Let us condition over the cost of all the edges of S_1 and of all the edges of S_2 but one. Then c_2 is a random variable which takes one value uniformly at random in a set of $2T^4$ distinct values. Hence the event $\{c_1 = c_2\}$ happens with probability at most $1/2T^4$.

Consider now any three lines $S_1(\lambda)$, $S_2(\lambda)$ and $S_3(\lambda)$. With the same notation as above, these three lines intersect at the same point if and only if

$$\begin{cases} c_1 + \lambda(\ell_1 - L) = c_2 + \lambda(\ell_2 - L) \\ c_1 + \lambda(\ell_1 - L) = c_3 + \lambda(\ell_3 - L) \end{cases}$$

If $\ell_1 = \ell_2$, it must be $c_1 = c_2$ which happens with probability at most $1/2T^4$ by the same argument as before. Otherwise it must be $c_3 = c_1 + \lambda(\ell_1 - \ell_3)$ with $\lambda = (c_1 - c_2)/(\ell_2 - \ell_1)$: this happens with probability at most $1/2T^4$ by a similar argument.

Since there are at most T^2 pairs and T^3 triples of the kind above, by the union bound the probability that the property of the claim is not satisfied is at most $T^2/(2T^4) + T^3/(2T^4) \leq 1/T$. □

Lemma 4.2 immediately implies that, with high probability, there are exactly two solutions of optimal Lagrangian cost. This will be crucial to prove that the running time of `bmst` is polynomial in expectation.

4.2 Analysis The following lemma shows approximation guarantee and efficiency of algorithm `bmst`.

LEMMA 4.3. *For any fixed $\epsilon \in (0, 1]$, Algorithm* `bmst` *is $(1+5\epsilon)$-approximate and its expected running time is polynomial.*

Proof. The perturbation can be performed in $O(m \log(2T^4)) = O(m^2)$ time. Algorithm `bmst` runs $O((m \log_{1+\epsilon} \frac{c_{max}}{c_{min}})^{1/\epsilon})$ instances of `lagrangian`, which is polynomial in the input size for any constant $\epsilon > 0$. The running time of `lagrangian` is dominated, modulo polynomial factors, by the number of spanning trees of optimal Lagrangian cost for the instance considered. This number is at most T deterministically, and it is exactly 2 with probability at least $(1 - 1/T)$ by Lemma 4.2. Hence the expected running time of `lagrangian` is polynomial.

Let OPT be the optimum solution to the perturbed instance, and F the $1/\epsilon$ most expensive edges in OPT. Consider the assignment $g : F \to \{c_1, \ldots, c_q\}$ such that, for each $e \in F$, $g(e) \geq c(e) \geq g(e)/(1 + \epsilon)$. Let OPT' and OPT'' be the optimum solution to the subproblems induced by $j = (F, g(\cdot))$ and $(F, c|_F(\cdot))$, respectively. Observe that $c(OPT'') =$

$c(OPT) - c(F)$. Moreover, $c(OPT') \leq c(OPT'')$ since $\min_{e\in F}\{c(e)\} \leq \min_{e\in F}\{g(e)\}$ (intuitively, for a given guess F we prune more edges with $c(\cdot)$ than with $g(\cdot)$). Altogether $c(OPT') \leq c(OPT) - c(F)$. Note also that in subproblem $\mathcal{P}_j$ each edge e costs at most $\epsilon(1+\epsilon)c(OPT)$. If $\lambda^* = 0$, $c(S^-) = c(OPT')$. Otherwise

$$\begin{aligned} c(S^-) &= c(S^+) + c(e^-) - c(e^+) \leq LAG(\lambda^*) + c(e^-) \\ &\leq c(OPT') + c(e^-) \\ &\leq c(OPT') + \epsilon(1+\epsilon)c(OPT). \end{aligned}$$

It follows that $c(F)+c(S^-) \leq c(OPT)+\epsilon(1+\epsilon)c(OPT)$. It is easy to see that the initial perturbation increases the cost of the optimal solution at most by a factor $(1+\epsilon)$ (deterministically). Thus the overall approximation factor is $(1+\epsilon)(1+\epsilon(1+\epsilon)) \leq 1+5\epsilon$. □

Before presenting the proof of the bitonicity of algorithm `lagrangian`, let us give some intuitions about bitonicity in case (i) of Definition 2.1; the other case is similar. Let $(\lambda^*, LAG(\lambda^*))$ and $(\bar{\lambda}^*, \overline{LAG}(\bar{\lambda}^*))$ be the optimal Lagrangian point in the original and modified problems respectively.

A high level intuition in this case is that when an agent f decreases her $\ell(f)$ or her $c(f)$ then the original output line S^- either rotates down (with its c-point fixed) or translates down. Because of concavity of $LAG(\lambda^*)$ this can move λ^* only to the left, thus $\bar{\lambda}^* \leq \lambda^*$. This implies monotonicity, because all non-positive slope lines in the modified problem intersecting $(\bar{\lambda}^*, \overline{LAG}(\bar{\lambda}^*))$ must contain f.

For bitonicity, consider the optimal Lagrangian multiplier λ^* and recall that we output a line S^- adjacent to line S^+, where both lines intersect $(\lambda^*, LAG(\lambda^*))$, and $c(S^-)$ is highest among such lines S^- . We have two main different cases: $\bar{\lambda}^* = \lambda^*$ and $\bar{\lambda}^* < \lambda^*$. Since we consider that $f \in S^-$ and in case $\bar{\lambda}^* = \lambda^*$ it must be the case that also some positive slope line, say S', with $(\lambda^*, LAG(\lambda^*)) \in S'$, contains f, and so $\bar{S}'$ will belong to the new optimal point $(\bar{\lambda}^*, \overline{LAG}(\bar{\lambda}^*))$. Now, the lines S^+ from $(\bar{\lambda}^*, \overline{LAG}(\bar{\lambda}^*))$, such that $f \notin S^+$, are not translated to $(\bar{\lambda}^*, \overline{LAG}(\bar{\lambda}^*))$, and thus we lose some of the previous adjacent pairs (S^-, S^+). But, the new output such adjacent pair $(\bar{S}^-, \bar{S}^+)$ can only have smaller value of $c(\bar{S}^-)$ (because in the original problem $(\lambda^*, LAG(\lambda^*))$ we output such line with maximum $c(S^-)$). In the second case when $\bar{\lambda}^* < \lambda^*$, we can observe that there can be some new non-positive slope lines in $(\bar{\lambda}^*, \overline{LAG}(\bar{\lambda}^*))$, but their c-cost can only be lower than the previous $c(S^-)$. This follows by the concavity of LAG.

LEMMA 4.4. *Algorithm* `lagrangian` *is bitonic with respect to* $c(\cdot)$.

Proof. By the concavity of LAG, if $\lambda^* = 0$, then observe that $\bar{\lambda}^* = 0$ and the solution returned in the original and modified problems is exactly the same. On the other hand, for $\lambda^* = +\infty$ no solution is returned for the original problem. We note also that if cost and length of f are unmodified then the algorithm returns the same solution in the original and modified problems. In all these cases `lagrangian` is trivially bitonic. Hence assume that $0 < \lambda^* < +\infty$ and that f modifies either cost or length. We distinguish two cases:

(i) Case $f \in S^-$: We have to show that $f \in \bar{S}^-$ and $\bar{c}(\bar{S}^-) \leq c(S^-)$. Note that $\bar{c}_{\lambda^*}(S^-)$ is either equal to $c_{\lambda^*}(S^-) - \Delta < \bar{c}_{\lambda^*}(S^-)$ or to $c_{\lambda^*}(S^-) - \lambda^*\Delta < \bar{c}_{\lambda^*}(S^-)$, depending on whether the cost or length of f is modified (decreased) by Δ, respectively. For any $S \not\ni f$, $\bar{c}_{\lambda^*}(S) = c_{\lambda^*}(S) \geq c_{\lambda^*}(S^-)$. (The last inequality follows from the fact that point $(\lambda^*, c_{\lambda^*}(S^-))$ belongs to the lower envelope LAG and that the lower envelope is defined as the point-wise minimum of all lines $c_\lambda(S)$ for all spanning trees S.) Hence $\overline{LAG}$ intersects at λ^* only solutions containing f in the modified problem: since S^- is one of those solutions, the concavity of $\overline{LAG}$ implies that $\bar{\lambda}^* \leq \lambda^*$.

Suppose first $\bar{\lambda}^* = \lambda^*$. Note that in this case all the solutions intersecting $\overline{LAG}$ at λ^* contain f, and hence $f \in \bar{S}^-$. Moreover, those solutions are exactly the solutions containing f which intersect LAG at the same value of the Lagrangian multiplier as in the original problem. By construction $\bar{S}^-$ is adjacent to a positive-slope solution $\bar{S}^+$ intersecting $\overline{LAG}$ at λ^*. It follows that (S_0^+, S_0^-) is a candidate pair for the original problem as well, where lines (S_0^+, S_0^-) correspond to lines $(\bar{S}^+, \bar{S}^-)$ for the original (unmodified) cost/length of f. The maximality of S^- implies $c(S^-) \geq c(S_0^-) \geq c(\bar{S}^-) \geq \bar{c}(\bar{S}^-)$.

Suppose now $\bar{\lambda}^* < \lambda^*$. Under this assumption, $S^+ \not\ni f$, since otherwise we would have $\bar{\lambda}^* \geq \lambda^*$ by the same argument as before. This implies that the lower envelope $LAG_{\not f}$ of the lines not containing f intersects S^+ at λ^* (since $LAG_{\not f}$ is sandwiched between LAG and S^+). It follows from the concavity of $LAG_{\not f}$ that $LAG_{\not f}$ is defined only by positive-slope lines for $\lambda < \lambda^*$, and consequently the (non-positive-slope) solution $\bar{S}^-$ returned for the modified problem must contain f. Since $\bar{S}^-$ intersects $\overline{LAG}$ at $\bar{\lambda}^* < \lambda^*$ in the modified problem, the slope of $\bar{S}^-$ is larger than the slope of S^- (in both problems), unless S^- and $\bar{S}^-$ correspond to the same spanning tree. In the former case, we can conclude by Lemma 4.1, applied to the lower envelope $\overline{LAG}_f$ of the lines containing f in the modified problem, that $\bar{c}(\bar{S}^-) \leq \bar{c}(S^-) \leq c(S^-)$. And if S^- and $\bar{S}^-$ correspond to the same spanning tree, we have $\bar{c}(\bar{S}^-) = \bar{c}(S^-) \leq c(S^-)$.

(ii) Case $f \notin S^-$: If $f \in \bar{S}^-$, there is nothing to show. So assume $f \notin \bar{S}^-$: we have to show that $\bar{c}(\bar{S}^-) \geq c(S^-)$. For $\lambda > \lambda^*$, the solutions of non-positive slope not containing f have Lagrangian cost larger than $LAG(\lambda^*) \geq \overline{LAG}(\lambda^*) \geq \overline{LAG}(\bar{\lambda}^*)$. Therefore $\bar{\lambda}^* \geq \lambda^*$. We can conclude by the same argument as in Case (i) that the slope of $\bar{S}^-$ is not larger than the slope of S^-. Hence, by Lemma 4.1 applied to the lower envelope $LAG_{\not f} = \overline{LAG}_{\not f}$ of the lines not containing f, $\bar{c}(\bar{S}^-) = c(\bar{S}^-) \geq c(S^-)$. □

COROLLARY 4.1. *The procedure which returns $F_j \cup S_j$ for a given problem $\mathcal{P}_j$ is bitonic with respect to $c(\cdot)$.*

Proof. If $f \in F_j$, then $\mathcal{P}_j = \bar{\mathcal{P}}_j$ and hence $S_j = \bar{S}_j$. In this case the procedure is trivially bitonic. Otherwise ($f \notin F_j$), suppose $f \in F_j \cup S_j$ (i.e., $f \in S_j$). By Lemma 4.4, $f \in \bar{S}_j$ and $\bar{c}(\bar{S}_j) \leq c(S_j)$. It follows that $\bar{c}(F_j \cup \bar{S}_j) = c(F_j) + \bar{c}(\bar{S}_j) \leq c(F_j) + c(S_j) = c(F_j \cup S_j)$. It remains to consider the case $f \notin F_j \cup S_j$. If $f \in F_j \cup \bar{S}_j$ there is nothing to show. Hence assume $f \notin F_j \cup \bar{S}_j$, which implies $f \notin \bar{S}_j$. By Lemma 4.4, $\bar{c}(\bar{S}_j) \geq c(S_j)$ which implies $\bar{c}(F_j \cup \bar{S}_j) \geq c(F_j \cup S_j)$. □

LEMMA 4.5. *Algorithm* `bmst` *is monotone.*

Proof. Analogously to the proof of Lemma 3.3, we define a variant `ideal` of `bmst` which considers all the powers of $(1+\epsilon)$ for any guess. Also in this case the solution computed by `ideal` and `bmst` is the same. In fact, when $min_{e \in F_j}\{g_j(e)\} < c_{min}$, all the edges are removed and the problem becomes infeasible. Vice versa, for $min_{e \in F_j}\{g_j(e)\} > c_{max}$, no edge is discarded and one obtains exactly the same subproblem by replacing each $g_j(e)$ with $\min\{g_j(e), c_{max}\}$. Algorithm `ideal` is monotone by the Composition Theorem 2.1 and Corollary 4.1. It follows that `bmst` is monotone as well. □

Theorem 1.2 follows from Lemmas 4.3 and 4.5, by applying the results in [5, 15].

References

[1] A. Archer, C. H. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. *Internet Mathematics*, 1(2), 2003.

[2] F. Barahona and W. R. Pulleyblank. Exact arborescences, matchings and cycles. *Discrete Applied Mathematics*, 16(2):91–99, 1987.

[3] Y. Bartal, R. Gonen, and N. Nisan. Incentive compatible multi-unit combinatorial auctions. In *Proc. of TARK*, pages 72–87, 2003.

[4] D. Bilò, L. Gualà, and G. Proietti. Designing a truthful mechanism for a spanning arborescence bicriteria problem. In *Proc. of CAAN*, pages 19–30, 2006.

[5] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. In *Proc. of STOC*, pages 39–48, 2005.

[6] P. M. Camerini, G. Galbiati, and F. Maffioli. Random pseudo-polynomial algorithms for exact matroid intersection. *J. Algorithms*, 13:258–273, 1992.

[7] E. Clarke. Multipart pricing of public goods. *Public Choice*, 8:17–33, 1971.

[8] S. Dobzinski and N. Nisan. Mechanisms for multi-unit auctions. In *Proc. of EC*, pages 346–351, 2007.

[9] S. Dobzinski, N. Nisan, and M. Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proc. of STOC*, pages 644–652, 2006.

[10] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Computing*, 24(2):296–317, 1995.

[11] F. Grandoni. A randomized PTAS for the multi-budgeted matching problem. Manuscript.

[12] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[13] J. Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. *SIAM J. Computing*, 31(6):1783–1793, 2002.

[14] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proc. of FOCS*, pages 595–604, 2005.

[15] D. J. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49(5):577–602, 2002.

[16] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proc. of AAAI*, pages 379–384, 2002.

[17] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. In *Proc. of STOC*, pages 345–354, 1987.

[18] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. of STOC*, pages 129–140, 1999.

[19] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proc. of FOCS*, pages 86–92, 2000.

[20] R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem. In *Proc. of SWAT*, pages 66–75. 1996.

[21] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. H. III. Many birds with one stone: multi-objective approximation algorithms. In *Proc. of STOC*, pages 438–447, 1993.

[22] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[23] D. A. Spielman and S. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[24] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *J. Finance*, 16:8–37, 1961.