UTILITARIAN MECHANISM DESIGN FOR MULTI-OBJECTIVE OPTIMIZATION*

FABRIZIO GRANDONI†, PIOTR KRYSTA‡, STEFANO LEONARDI§, AND CARMINE VENTRE¶

Abstract. In a classic optimization problem the complete input data is assumed to be 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. In this setting we are given a 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. The main contribution of this paper is showing that two of the main tools for the design of approximation algorithms for multi-objective optimization problems, namely approximate Pareto sets and Lagrangian relaxation, can lead to truthful approximation schemes.

By exploiting the method of approximate Pareto sets, we devise truthful deterministic and randomized multi-criteria 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 construction also applies to *multi-dimensional knapsack* and *multi-unit combinatorial auctions*. Our FPTASs compute a $(1+\varepsilon)$-approximate solution violating each budget constraint by a factor $(1+\varepsilon)$.

When feasible solutions induce an independence system, i.e., when subsets of feasible solutions are feasible as well, we 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, where one wants to compute a minimum cost spanning tree whose length is at most a given value L. This result is based on the Lagrangian relaxation method, in combination with our monotone guessing step and with a random perturbation step (ensuring low expected running time). This result can be derandomized in the case of integral lengths.

All the mentioned results match the best known approximation ratios, which are however obtained by non-truthful algorithms.

Key words. algorithmic mechanism design, truthful mechanisms, multi-objective optimization, approximation algorithms, monotone algorithms, approximate Pareto sets, Lagrangian relaxation.

AMS subject classifications. 91A99, 91B26, 91B32, 68Q25, 68W25, 68W40, 90B10

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

*Partially supported by ERC Starting Grant NEWNET 279352, by DFG grant Kr 2332/1-2 within Emmy Noether program, by EPSRC grants EP/F069502/1 and EP/K01000X/1, by EU FET project MULTIPLEX 317532 and EU ERC project PAAI 259515. A preliminary version of this paper has appeared as [16].

†IDSIA, University of Lugano, Galleria 1, 6928 Manno-Lugano, Switzerland. Email: fabrizio@idsia.ch.

‡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

¶Teesside University, School of Computing, Greig Building, Borough Road, Middlesbrough TS1 3BA, U.K. Email: c.ventre@tees.ac.uk

assume $k = O(1)$. The aim is to find 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., [36] 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[1]. Let also $best \in \{\max, \min\}$ and $\succeq_i \in \{\geq, \leq\}$ for $i = 1, \ldots, k$. We consider a problem $\mathcal{P}$ defined as:

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

We will often refer to the $\ell_0(e)$'s as *costs*, and call *lengths* the remaining $\ell_i(e)$'s. We define $\succeq_0 = \geq$ if $best = \max$, and $\succeq_0 = \leq$ 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. (For basic notions on algorithmic game theory, see, e.g., [31]). 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 BMST example above, agent f might declare a larger delay $\ell'_1(f)$ 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 to force f to declare her true type. This is achieved by defining a set of *payments* p^f, $f \in \mathcal{U}$. Payments, types and the computed solution define agents' *utility* (see Section 2 for formal definitions). 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] Part of our results can be extended to the case that zero values of the $\ell_i(e)$'s and B_i's are also allowed. This however involves more technical algorithms and analysis, without introducing any substantially new idea.

1.1. Related Work. A classical technique to design truthful mechanisms is the Vickrey-Clarke-Groves (VCG) mechanism [10, 20, 39] which however requires to solve optimally the underlying optimization problem. For NP-hard problems one can use the approach of [7, 24]; therein, it is shown 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 $i \in \{0, 1, \ldots, k\}$, $\ell_i(f)$ with $\bar{\ell}_i(f) \succeq_i \ell_i(f)$[2].

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 [7], where B is the smallest budget. Their result extends to *multi-unit combinatorial auctions* for unknown single-minded bidders (i.e., the setting in which bidders are interested in only one set of goods and can lie on both valuation and set itself). (See [3, 12, 13, 23] for results on more general types of bidders.) A monotone PTAS is known for *multiple knapsack* [7], 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 [7]. 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 [5].

Multi-budgeted optimization in the standard (non-game-theoretical) sense is extensively studied in the literature. There are a few general tools for designing approximation algorithms for budgeted problems. One basic approach is combining *dynamic programming* (which solves the problem for polynomial costs and lengths) with *rounding and scaling* techniques (to reduce the problem to the case of polynomial quantities). This leads for example to an FPTAS for BSP [21, 25, 40].

Another fundamental technique is the *Lagrangian relaxation method*. The basic idea is relaxing the budget constraints, and lifting them into the objective function, where they are weighted by Lagrangian multipliers. Solving the relaxed problem, one obtains two or more solutions with optimal Lagrangian weight, which can—if needed—be patched together to get a good solution for the original problem. Demonstrating this method, Goemans and Ravi [35] gave a PTAS for BMST, which also extends to 1-budgeted matroid basis. Inspired by this approach, Correa and Levin [11] presented algorithms for special classes of polynomial-time covering problems with an additional covering constraint. Using the same approach as Goemans and Ravi, with an involved patching step, Berger, Bonifaci, Grandoni, and Schäfer [4] obtained a PTAS for 1-budgeted matching and 1-budgeted matroid intersection independent set. A PTAS for 2-budgeted matching was obtained by Grandoni and Zenklusen [18, 19]. The authors also present a PTAS for k-budgeted matroid independent set, $k = O(1)$. Finally a PTAS for k-budgeted matching was given by Chekuri, Vondrák, and Zenklusen [9].

[2]Sometimes in the literature a weaker notion of monotonicity is considered, where only the cost $\ell_0(f)$ 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 [15]. 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 parameters $\ell_i(f)$'s.

The authors also present a PRAS for k-budgeted matroid intersection[3].

The mentioned algorithms provide strictly feasible solutions. Sometimes one searches for *multi-criteria* PTASs, which compute a $(1+\varepsilon)$-approximate solution violating the budgets by a factor $(1+\varepsilon)$. Papadimitriou and Yannakakis [32] describe a very general technique, based on the construction of ε-approximate Pareto sets (see Section 1.2 for more details about this technique). Their approach provides multi-criteria FPTASs for k-budgeted spanning tree and k-budgeted shortest path. Furthermore, it implies a multi-criteria FPRAS for k-budgeted (perfect) matching. Grandoni and Zenklusen [18, 19] describe a simple technique to convert multi-criteria PTASs/PRASs into *pure* PTASs/PRASs, with no violation of the budget constrains, in the case that feasible solutions (satisfying also the budget constraints) induce an *independence system*. We recall that an independence system is a set system $(\mathcal{U},\mathcal{S})$ with the extra property that for any $S\in\mathcal{S}$ and $S'\subseteq S$, one has $S'\in\mathcal{S}$ (namely, subsets of feasible solutions are feasible as well). This methods leads for example to a PRAS for k-budgeted matching.

Grandoni, Ravi, and Singh [17, 18] show how to apply the *iterative randomized rounding* technique to multi-budgeted optimization problems. They obtain a multi-criteria PTAS for k-budgeted spanning tree and k-budgeted matroid basis where budgets are violated by a factor $(1+\varepsilon)$ and the cost of the solution matches the optimal cost. They also obtain a multi-criteria PTAS for k-budgeted bipartite matching, which can be converted into a PTAS using the mentioned technique in [18, 19].

All mentioned problems are easy in the unbudgeted version. Given an NP-hard unbudgeted problem which admits a ρ-approximation, the *parametric search* technique in [26] provides a multi-criteria $k\rho$-approximation algorithm violating each budget by a factor $k\rho$ for the corresponding problem with k budgets. Other techniques lead to logarithmic approximation factors (see, e.g., [6, 34]).

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 sets and Lagrangian relaxation, can lead to monotone algorithms and thus to truthful mechanisms. When monotone algorithms are randomized the notion of truthfulness changes according to how the outcome of the random coin tosses influences the utilities; we focus here on the notions of *probabilistic* and *universal truthfulness*. We recall that a randomized mechanism is *probabilistically truthful* if saying the truth is a dominant strategy with high probability (see, e.g., [1]); a randomized mechanism is, instead, *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 [13, 14, 30].

Monotone Construction of Approximate Pareto Sets. Our first contribution is a family of monotone multi-criteria FPTASs for the optimization problems $\mathcal{P}$ 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 of the original one (in this case we say that e is *discarded*). This is trivially true for all the applications considered in this paper.

[3] A PRAS is a randomized PTAS, i.e., a PTAS which produces the desired solution with probability at least 1/2. FPRASs are defined analogously w.r.t. FPTASs.

For example, in the BMST case it is sufficient to delete edge e from the graph. Our approximation schemes compute, for any given $\varepsilon > 0$, a solution within a factor $(1+\varepsilon)$ of the optimum which violates each budget constraint by a factor of at most $(1+\varepsilon)$. The running time is polynomial in the size of the input and $1/\varepsilon$.

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

Our result implies truthful multi-criteria FPTASs for a number of natural problems. For example this covers the mentioned BMST and BST problems. One can also handle the generalizations of BMST and BSP 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 [7]. 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 [7] (see Section 1.1).

In the case of multi-budgeted (perfect) matching, where $\mathcal{S}$ is the set of (perfect) matchings of a given graph, the unique pseudo-polynomial-time exact algorithm known is Monte-Carlo [28] (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. 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 [8] implies a probabilistically truthful FPTAS for the multi-budgeted version of the problem of finding a basis in the intersection of two matroids.

Our approach builds up on the construction of approximate Pareto sets by Papadimitriou and Yannakakis [32]. More precisely, it exploits a sufficient condition given by the authors for the efficient construction of such approximate Pareto sets. In more detail, 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 set. The authors of [32] introduce the notion of ε-approximate Pareto set, i.e., a subset of solutions such that every Pareto solution is within a factor $(1+\varepsilon)$ on each objective from some solution in the mentioned subset. They prove that there is always an ε-approximate Pareto set of polynomial size (in the size of the problem and $1/\varepsilon$). Intuitively (and with the notation of this paper), consider the $(k+1)$-dimensional hyperrectangle induced by the range of solutions costs and lengths. Subdivide this hyperrectangle into (polynomially many) sub-hyperrectangles using powers of $(1+\varepsilon)$ in a standard way. It is sufficient to include in the approximate Pareto set one solution in each sub-hyperrectangle, if any (dominated solutions can be filtered out at the end of the process). Equivalently, it is sufficient to solve polynomially many instances of the following *gap problem*: given a collection of target values B'_i, $i = 0, \ldots, k$, either output a solution S with $\ell_i(S) \succeq_i B'_i$ for all i, or prove that there is no solution S such that $\ell_i(S) \succeq_i B''_i$ for all i, where $B''_i = B'_i(1+\varepsilon)$ for $\succeq_i = \geq$ and $B''_i = B'_i/(1+\varepsilon)$ otherwise.

In the same paper the authors show that a sufficient condition to solve one gap problem in polynomial time (in the size of the problem and $1/\varepsilon$) is the existence of a pseudo-polynomial-time algorithm $\mathcal{A}$ for the exact version of the problem. The basic

idea is to round costs, lengths, and target values so that they range in a polynomial-size set. The rounding is such that the gap problem in the original instance is equivalent to a feasibility problem in the rounded instance. It is easy to encode costs and lengths in a unique weight function (with polynomially many possible values). Then it is sufficient to guess the weight of some feasible solution, and use $\mathcal{A}$ to find a solution of exactly that target weight. Their approach implies a multi-criteria FPTAS for the associated optimization problems where all the objectives but one are turned into budget constraints, i.e., for our framework: it is sufficient to explore the portion of the ε-approximate Pareto set corresponding to almost feasible solutions, and output the best such solution[4].

In this paper we show how to make monotone the multi-criteria FPTAS above. From the above discussion, we need to solve a few gap subproblems, and select the best obtained solution. 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 [7, 27], 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. The authors of [7] show that a proper composition of bitonic algorithms is monotone and we use the same basic approach. One extra issue that we need to address here is that the set of solved subproblems changes when some value $\ell_0(e)$ is modified, so that the composition of bitonic algorithms in [7, 27] cannot be directly applied. We address this issue by expanding the set of subproblems suitably, and proving that the resulting algorithm is equivalent to an *ideal* algorithm which solved infinitely many subproblems; the composition theorem in [7, 27] can then be applied to this ideal algorithm.

Monotone Lagrangian Relaxation. The approach above relies on the existence of a pseudo-polynomial-time exact algorithm and violates budget constraints by a $(1+\varepsilon)$ 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 Las Vegas[5] PTAS for BMST. This implies a universally truthful mechanism for the corresponding problem (see Section 4). We are able to derandomize our result in the case of integer lengths. It remains open whether the same result is possible also for arbitrary lengths[6].

THEOREM 1.2. *There is a universally truthful Las Vegas PTAS for BMST. In the special case that edge lengths are integral, there is a deterministic truthful PTAS for BMST.* For a comparison, in [7] a truthful FPTAS is given for the same problem on bounded-treewidth graphs, while the deterministic PTAS in [35] is not truthful.

The basic idea is combining our framework based on the composition of bitonic algorithms with the (non-monotone) PTAS of Ravi and Goemans [35]. Roughly speaking, the PTAS in [35] works as follows (see also, e.g., [22, 29] and references therein for the Lagrangian relaxation method). Recall that we search for a spanning tree S^* of minimum cost $\ell_0(S^*)$ so that its length $\ell_1(S^*)$ is at most a given value L. For a

[4]We remark that also the converse is true: a multi-criteria FPTAS can be used to solve the gap problem, hence to construct the approximate Pareto set.

[5]We recall that a Las Vegas polynomial time algorithm is an algorithm whose expected running time is polynomial.

[6]This question has mostly a theoretical flavor: in most applications assuming integer lengths (and even integer costs) is reasonable. The case of a bounded number of decimals can be easily reduced to the integer case.

spanning tree S, consider the Lagrangian cost $c_\lambda(S) = \ell_0(S) + \lambda(\ell_1(S) - L)$, where $\lambda \geq 0$ is the *Lagrangian multiplier*. A standard polynomial-time procedure computes the maximum value $LAG(\lambda^*)$ of $c_\lambda(S)$ over the possible values of λ and S. Let λ^* be the value of λ achieving this maximum. Observe that $LAG(\lambda^*) \leq \ell_0(S^*)$. The same procedure also finds two spanning trees S^- and S^+, with $\ell_1(S^-) \leq L < \ell_1(S^+)$, of optimal Lagrangian cost $c_{\lambda^*}(S^-) = c_{\lambda^*}(S^+) = LAG(\lambda^*)$. By swapping edges in a careful way for polynomially many times, one can also enforce S^- and S^+ to be adjacent, i.e., one tree can be obtained from the other by swapping one edge. By construction, $\ell_0(S^+) \leq c_{\lambda^*}(S^+) = LAG(\lambda^*) \leq \ell_0(S^*)$, hence $\ell_0(S^-) \leq \ell_0(S^*) + \ell_{0,max}$ where $\ell_{0,max}$ is the maximum cost of one edge. In order to get a PTAS, Ravi and Goemans simply guess the $1/\varepsilon$ most expensive edges of the optimal solution in a preliminary step, and reduce the problem consequently (so that $\ell_{0,max}$ introduces a small factor in the approximation)[7].

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 unfeasible 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. Indeed, there might be exponentially many pairs (S^-, S^+) which satisfy the conditions of Ravi and Goemans's approach. This introduces symmetries which have to be broken in order to guarantee bitonicity. One possibility is choosing, among the candidate pairs (S^-, S^+), the one with largest $\ell_0(S^-)$. 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 large. We solve this problem by means of randomization, in a way pretty close to the smoothed analysis of (standard) algorithms [38]. 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 sufficiently high probability there will be only one candidate pair (S^-, S^+). 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

[7] The PTAS in [35] is presented to return an unfeasible 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.)

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 [35]. This analysis differs from the standard packing LP sensitivity analysis (see, e.g., [37]) in that we also change an entry of the LP matrix and not only the right-hand-side constants.

In the case of integer lengths, we first round the costs to make them integral, and then add a proper deterministic perturbation to each cost. This way, we guarantee that there is at most one candidate pair (S^-, S^+) (hence the running time is polynomial).

Independence Systems. Consider any problem $\mathcal{P}$ of the type considered in Theorem 1.1, with the further constraint that its solution space induces an *independence system*, i.e., such that removing any element from a feasible solution preserves feasibility. We remark that by solution space we mean the solutions $S \in \mathcal{S}$ which *also* satisfy the budget constraints. We also assume that we can remove in polynomial time all the solutions in $\mathcal{S}$ *not* containing a given element $e \in \mathcal{U}$ (in this case we say that e is *selected*). This family of problems includes, for example, the problem of computing a (possibly non-perfect) matching of maximum cost, subject to budget upper bounds. In that case the selection of an edge e can be obtained by deleting e and all the edges incident to it, and decreasing by $\ell_i(e)$ each budget B_i. Given a solution for the reduced problem, the corresponding solution for the original problem is obtained by adding back e.

We can combine the monotone guessing step above with a variant[8] of the truthful FPTAS from Theorem 1.1 to obtain truthful PTASs for the mentioned problems. The proof of the following theorem appears in Section 5.

THEOREM 1.3. *There is a (probabilistically) truthful PTAS for multi-objective problems $\mathcal{P}$ whose feasible solutions induce an independence system and which admit 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 [7]. For a given algorithm $\mathcal{A}$, let S' be the solution given in output by $\mathcal{A}$ on input declarations $(\ell'^e)_{e\in\mathcal{U}}$. Agent f with type ℓ^f evaluates solution S' according to the following valuation function:

(2.1)
$$v^f(S', \ell^f) = \begin{cases} \ell_0(f) & \text{if } best = \max, f \in S' \text{ and } \ell'_i(f) \preceq_i \ell_i(f) \text{ for } i = 1, \ldots, k; \\ -\ell_0(f) & \text{if } best = \min, f \in S' \text{ and } \ell'_i(f) \preceq_i \ell_i(f) \text{ for } i = 1, \ldots, k; \\ -\infty & \text{if } f \in S' \text{ and } \ell'_i(f) \succ_i \ell_i(f) \text{ for some } i \in \{1, \ldots, k\}; \\ 0 & \text{otherwise.} \end{cases}$$

A *mechanism* is an algorithm $\mathcal{A}$ together with a payment function p. The mechanism $(\mathcal{A}, p)$ on input $(\ell'^e)_{e\in\mathcal{U}}$ computes a solution S' and awards agent f a finite payment $p^f((\ell'^e)_{e\in\mathcal{U}})$ which is non-positive if $best = \max$ and non-negative when $best = \min$. Agent f derives *utility*

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

[8] A similar approach is used to obtain a (non-monotone) PRAS for the k-budgeted matching problem in [18, 19].

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

$$u^f((\ell^f, \ell'^{-f}), \ell^f) \geq u^f((\ell'^f, \ell'^{-f}), \ell^f)$$

for each ℓ'^f and ℓ'^{-f}, with ℓ'^{-f} denoting the declarations of all agents but f.

The following definition of *monotonicity* of an algorithm turns out to be very useful in the design of truthful mechanisms.

DEFINITION 2.1. *Let S and $\overline{S}$ be the solutions computed by an algorithm $\mathcal{A}$ with declarations $(\ell^e)_{e\in\mathcal{U}}$ and $(\bar{\ell}^e)_{e\in\mathcal{U}}$, respectively. Algorithm $\mathcal{A}$ is* monotone *if $f \in S$ implies that $f \in \bar{S}$ whenever $\ell^{-f} = \bar{\ell}^{-f}$ and $\bar{\ell}_i(f) \succeq_i \ell_i(f)$ for all $i \in \{0, 1, \ldots, k\}$.*

From the previous definition, in order to prove the monotonicity of an algorithm, it is sufficient to consider two declarations which are identical apart for exactly one entry where $\bar{\ell}'_i(f) \succ_i \ell'_i(f)$.

In [7, 24] it is shown that designing a *monotone* algorithm is sufficient to obtain a truthful mechanism for agents with valuation and utility functions as above.

LEMMA 2.2 ([7, 24]). *For generalized single-minded agents, a monotone algorithm $\mathcal{A}$ admits a payment function p such that $(\mathcal{A}, p)$ is a truthful mechanism for $\mathcal{P}$.*

We only outline here how monotone algorithms imply the existence of payment functions, and thus of truthful mechanisms. For more details, please refer to [7, 24]. For the sake of simplicity, let us consider a standard (single-objective) maximization problem. In particular, $k = 0$ here. Let $\mathcal{A}$ be a given monotone algorithm, and consider a given agent f and a fixed value of ℓ'^{-f}. The monotonicity of $\mathcal{A}$ implies the existence of a critical value $\theta_f^{\mathcal{A}}$ (depending on ℓ'^{-f}), such that f is included in the solution S output by $\mathcal{A}$ (f is winning) if $\ell'_0(f) \geq \theta_f^{\mathcal{A}}$, and f is not included in S (f is not winning) otherwise. Consider the mechanism where the payment for a winning agent f is the critical value $\theta_f^{\mathcal{A}}$ (given the declarations ℓ'^{-f} of the remaining agents), and the payment is zero if f is not winning. Note that critical values can be computed by performing a binary search on the possible values $\ell''_0(f)$ of $\ell_0(f)$ for each agent f, and running $\mathcal{A}$ with declarations $(\ell'^{-f}, \ell''_0(f))$. In [7, 24] it is shown that such mechanism is truthful[9].

The combination (e.g., taking the best solution) of monotone algorithms is not necessarily monotone. To gain some intuition, consider the following simple example given in [27]. We wish to sell a single good to a set of three agents $\{1, 2, 3\}$. Let $\ell'_0(i) =: v_i$ be the value declared by agent i for the mentioned good. The objective function is the value v_i of the agent i who receives the good (0 if the good is not assigned). Consider the following two algorithms. Algorithm $\mathcal{A}_1$ assigns the good to bidder 2 if $v_1 \leq 1$, assigns the good to bidder 3 if $1 < v_1 \leq 2$, and otherwise assigns the good to bidder 1. Algorithm $\mathcal{A}_2$ instead does not assign the good if $v_1 < 1/2$, and assigns the good to bidder 1 if $v_1 \geq 1/2$ (bidders 2 and 3 never receive the good).

It is not hard to check that these two algorithms are monotone. Consider first $\mathcal{A}_1$. The critical value for bidder 1 is $\theta_1^{\mathcal{A}_1} = 2$, independently from the declarations of the other bidders. The critical values $\theta_2^{\mathcal{A}_1}$ and $\theta_3^{\mathcal{A}_1}$ are either 0 or $+\infty$, depending on v_1. For algorithm $\mathcal{A}_2$, one has $\theta_1^{\mathcal{A}_2} = 1/2$ and $\theta_2^{\mathcal{A}_2} = \theta_3^{\mathcal{A}_2} = +\infty$, independently from the declarations of the other bidders. Consider next the algorithm $\mathcal{A}_3$ which simply runs $\mathcal{A}_1$ and $\mathcal{A}_2$, and selects the solution which maximizes the objective value. It turns our that $\mathcal{A}_3$ is not monotone. To see that, fix $v_2 = 1/2$ and $v_3 = 3/2$. Then,

[9] The results in [7, 24] also require the algorithm to be *exact*. In combinatorial auctions terminology, an allocation algorithm is exact if it allocates to each bidder the declared set or nothing. However, in our context any algorithm for $\mathcal{P}$ is exact due to the constraint $x_e \in \{0, 1\}$.

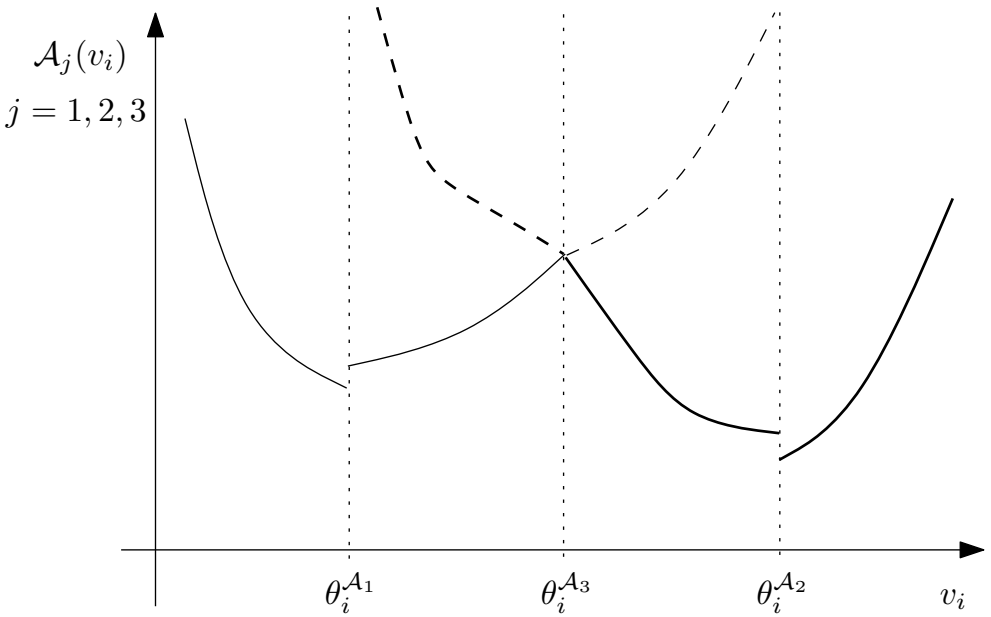

FIG. 1. *Consider two bitonic algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. The thin (resp., thick) curves represents the objective value $\mathcal{A}_1(v_i)$ (resp., $\mathcal{A}_2(v_i)$) returned by $\mathcal{A}_1$ (resp., $\mathcal{A}_2$) as a function of the declaration v_i of agent i (assuming that the other declarations are fixed to some value). Consider the algorithm $\mathcal{A}_3$ which outputs the solution of largest value among the ones output by $\mathcal{A}_1$ and $\mathcal{A}_2$. The corresponding objective value $\mathcal{A}_3(v_i)$ is given by the dashed curve.*

$\mathcal{A}_3$ does not assign the good to agent 1 if $v_1 \in [0, 1/2) \cup (1, 3/2)$, and does assign the good to agent 1 if $v_1 \in (1/2, 1)$. Hence the monotonicity property is not satisfied.

To handle cases like the one above, the following stronger notion of *bitonicity* turns out to be useful. Intuitively, for a (single-objective) maximization problem, a monotone algorithm $\mathcal{A}$ is bitonic (w.r.t. the objective function) if, for each agent f and fixed ℓ'^{-f}, the objective value of the solution found by $\mathcal{A}$ is a non-increasing function of $\ell_0(f)$ for $\ell_0(f) < \theta_f^{\mathcal{A}}$, and it is a non-decreasing function of $\ell_0(f)$ for $\ell_0(f) \geq \theta_f^{\mathcal{A}}$. This property is sufficient to guarantee a monotone (bitonic, in fact) composition of algorithms, see Figure 1 for an intuition. In the above example, $\mathcal{A}_2$ is bitonic. On the other hand, $\mathcal{A}_1$ is not bitonic: e.g., if $v_2 < v_3$, at value $v_1 = 1 < \theta_1^{\mathcal{A}_1}$ the objective function increases. We remark that bitonicity can be also defined w.r.t. some function different from the objective function.

We now apply bitonicity to our setting. Consider a *cost function* $c : 2^{\mathcal{U}} \to \mathbb{Q}^+$. Intuitively, c plays the same role as the objective function in the previous examples. The choice of c will depend on the context.

DEFINITION 2.3. *Let S be the solution computed by an algorithm $\mathcal{A}$ with declarations $(\ell^e)_{e\in\mathcal{U}}$. For a vector $(\bar{\ell}^e)_{e\in\mathcal{U}}$, with $\ell^{-f} = \bar{\ell}^{-f}$ and $\bar{\ell}_i(f) \succeq_i \ell_i(f)$ for all $i \in \{0, 1, \ldots, k\}$, we let $\overline{S}$ denote the solution computed by $\mathcal{A}$ on input $(\bar{\ell}^e)_{e\in\mathcal{U}}$. Algorithm $\mathcal{A}$ is* bitonic *with respect to a cost function $c(\cdot)$ if the following properties hold:*

(i) If $f \in S$, then $f \in \bar{S}$ and $\bar{c}(\bar{S}) \succeq_0 c(S)$.
(ii) If $f \notin S$, then either $f \in \bar{S}$ or $\bar{c}(\bar{S}) \preceq_0 c(S)$.

Observe that, for a maximization problem ($\succeq_0 = \geq$), this definition precisely means

that when agent f is not winning ($f \notin S$ and $f \notin \bar{S}$) then the cost $c(\cdot)$ of the solution is non-increasing (as function of $\ell_0(f)$), and when agent f is winning ($f \in S$ and $f \in \bar{S}$) then the cost $c(\cdot)$ of the solution is non-decreasing.

In the following, we usually let $\mathcal{P}$ be the original problem and $\bar{\mathcal{P}}$ denote a problem in which f changes its declaration as in the above definition. Observe that a bitonic algorithm is monotone, while the vice versa is not necessarily true. As shown in [7], bitonic algorithms can be combined to get a monotone algorithm.

THEOREM 2.4. (COMPOSITION THEOREM) [7] *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.*

We remark that, although no implementable algorithm produces an infinite family of subproblems, we will need the technicality of infinitely many subproblems in the proof of a few lemmas. For more details concerning this aspect we refer the reader to [7].

In the following, we will denote by f the element whose type is modified according to the definition of monotonicity/bitonicity, and by e a generic element. We use an upper bar to denote quantities in the modified problem. By $\mathcal{N}$ we denote a dummy (null) solution which is returned when no feasible solution exists. For notational convenience, we will assume that $\mathcal{N}$ does not contain any element, and that it has cost $-\infty$ for *best* = max and $+\infty$ otherwise.

Some considerations about our model. Generalized single-minded agents adapt well to our setting. Consider again the case in which $\mathcal{P}$ models the 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 (2.1) connects with two different research areas in algorithmic mechanism design. Mechanisms with verification, introduced by [30] and further studied in [33] (see also references therein), 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 (2.1) expresses the valuation of single-minded bidders in a combinatorial auction (considered the paradigmatic problem in algorithmic mechanism design) using exact 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).

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 to be $\ell^{e_1} = (\varepsilon, \varepsilon)$,

$\ell^{e_2} = (0,0)$ and $\ell^{e_3} = (0,H)$ for some $0 < \varepsilon \leq L$ 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: $\bar{\ell}^{e_3} = (0,L)$. In this case the VCG mechanism would output the spanning tree $\{e_2, e_3\}$ (i.e., the minimum cost tree among the seemingly feasible ones) and pay agent e_3 an amount of $\varepsilon > 0$. 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 selected agents always have a strictly positive utility). It is enough to consider the same instance with $\ell^{e_1} = (M, \varepsilon)$ and $\ell^{e_2} = (1,0)$. Any M-approximation algorithm in input ℓ^{e_1}, ℓ^{e_2} and $\bar{\ell}^{e_3}$ must select the tree comprised of edges e_2 and e_3.

We conclude that generalized single-minded agents is a general framework motivated by combinatorial auctions that well encompasses the multi-objective optimization problems that we consider.

3. Exact Algorithms 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 $\mathcal{A}$. We focus here on the case that $\mathcal{A}$ is deterministic, the randomized case being analogous. For these problems we describe a monotone multi-criteria FPTAS `multi`. Recall that we assume that one can in polynomial time discard an element e in the sense described in the introduction. Our approach is inspired, approximation-wise, by the construction of ε-approximate Pareto sets in [32], and crucially exploits the combination of bitonic procedures to achieve monotonicity.

3.1. Algorithm. Algorithm `multi` is described in Figure 2.[10]. The basic idea is to generate 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 $B_{0,1} \prec_0 \ldots \prec_0 B_{0,q}$ to be all the (positive and/or negative) powers of $(1+\varepsilon)$ between proper boundary values. Intuitively, $B_{0,j}$ is an approximate guess of the cost of OPT. The case that OPT is empty, and hence has cost zero, is handled in Step M2. 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 $\sum_{e \in \mathcal{U}} \ell_0(e) x_e \succeq_0 B_{0,j}$.

For a given subproblem $\mathcal{P}_j$ with $B_0 := B_{0,j}$, procedure `feasible` computes an ε-feasible solution to $\mathcal{P}_j$, i.e., a solution $S_j \in \mathcal{S}$ such that each budget constraint is

[10] The interested reader might initially just focus on the case that $\succeq_i = \geq$ for all i, which is probably more intuitive given our notation.

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

M1: Let $B_{0,1} \prec_0 \ldots \prec_0 B_{0,q}$ be the powers of $(1+\varepsilon)$ between $\ell_{0,min}\frac{1}{m(1+\varepsilon)}$ and $m\,\ell_{0,max}\lceil\frac{m(1+\varepsilon)}{\varepsilon}\rceil$. For $j = 1, \ldots, q$, let $(S_j, \ell_{0,j}(\cdot)) =$ `feasible`$(\mathcal{P}_j, \varepsilon, B_{0,j})$.

M2: Return the solution $S^* = S_h \neq \mathcal{N}$ optimizing $\ell_{0,h}(S_h)$, the *best* solution with largest index h in case of ties. If no such solution exists, return $\emptyset$ if it is feasible, and otherwise $\mathcal{N}$.

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

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

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

F1b: If $\succeq_0 = \leq$: $B'_0 = \lceil\frac{m(1+\varepsilon)}{\varepsilon}\rceil$, $I_0 = \{0, 1, \ldots, B'_0\}$. 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 $\succeq_i = \geq$: $B'_i = \lceil\frac{m(1+\varepsilon)}{\varepsilon}\rceil$, $I_i = \{B'_i, B'_i+1, \ldots, m\,B'_i\}$. $\ell_i(e) := \min\{\ell_i(e), B_i\}$. Let $\ell'_i(e) = \lceil\frac{B'_i}{B_i}\ell_i(e)\rceil$.

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

F3: Let $M = m\lceil\frac{m(1+\varepsilon)}{\varepsilon}\rceil + 1$ 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 $(S^*_z, \frac{B_0}{B'_0}\ell'_0(\cdot))$.

FIG. 2. *Algorithm* `multi`.

violated at most by a factor $(1+\varepsilon)$. 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 $\Lambda(S_z) = z$ (here we exploit the fact that M is large enough compared to the quantities $\ell'_i(\cdot)$). The domain $\times_{i=0}^{k} I_i$ covers all the possible feasible length vectors. 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 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* $(1+\varepsilon)^2$*-approximate solution, violating each budget constraint at most by a factor* $(1+\varepsilon)$*. The running time of the algorithm is polynomial in the size of the input and* $1/\varepsilon$*.*

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 $O((m/\varepsilon)^k) = O((m/\varepsilon)^{O(1)})$ since k is constant. Also, the fact that $k = O(1)$ implies that the values of $\varphi(\cdot)$ 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 $O((m/\varepsilon)^{O(1)})$ time. Altogether, the running time of `feasible` is $O((m/\varepsilon)^{O(1)})$. The number of subproblems generated by `multi` is $O(\log_{1+\varepsilon} \frac{m\,\ell_{0,max}}{\varepsilon \ell_{0,min}})$, which is polynomial in the size of the input and $1/\varepsilon$. The running time bound follows.

Consider now the approximation factor. We initially observe that the solution S^* returned by the algorithm violates each budget constraint at most by a factor $(1+\varepsilon)$. Indeed, consider any solution S returned for some subproblem $\mathcal{P}'_j$ and any $i \geq 1$. If $\succeq_i = \geq$, one has:

$$\ell_i(S) = \frac{B_i}{B'_i} \sum_{e \in S} \frac{B'_i}{B_i} \ell_i(e) \geq \frac{B_i}{B'_i} \sum_{e \in S} \left(\left\lceil \frac{B'_i}{B_i} \ell_i(e) \right\rceil - 1 \right) = \frac{B_i}{B'_i} \sum_{e \in S} (\ell'_i(e) - 1)$$
$$\geq B_i - \frac{B_i}{B'_i} m = B_i - \frac{B_i m}{\lceil m(1+\varepsilon)/\varepsilon \rceil} \geq B_i - \frac{B_i m}{m(1+\varepsilon)/\varepsilon} = \frac{B_i}{(1+\varepsilon)}.$$

Otherwise ($\succeq_i = \leq$),

$$\ell_i(S) = \frac{B_i}{B'_i} \sum_{e \in S} \frac{B'_i}{B_i} \ell_i(e) \leq \frac{B_i}{B'_i} \sum_{e \in S} \left(\left\lfloor \frac{B'_i}{B_i} \ell_i(e) \right\rfloor + 1 \right) = \frac{B_i}{B'_i} \sum_{e \in S} (\ell'_i(e) + 1)$$
$$\leq B_i + \frac{B_i}{B'_i} m = B_i + \frac{B_i m}{\lceil m/\varepsilon \rceil} \leq B_i + \frac{B_i m}{m/\varepsilon} = B_i(1+\varepsilon).$$

In both cases, in the second inequality we used the feasibility of S w.r.t. $\mathcal{P}'_j$.

We next show that $S_h := S^*$ is $(1+\varepsilon)^2$-approximate. The claim is trivially true if there is no feasible solution or the optimum solution is empty. Otherwise, let $OPT \neq \emptyset$ be an optimum solution to the original problem instance. Consider first the case that $\succeq_0 = \geq$ (i.e., we are considering a maximization problem). Recall that $B_{0,1} < \ldots < B_{0,q}$ are powers of $(1+\varepsilon)$. Let j be the largest index such that $\ell_0(OPT) \geq B_{0,j}(1+\varepsilon)$. In particular, $\ell_0(OPT) < B_{0,j+1}(1+\varepsilon) = B_{0,j}(1+\varepsilon)^2$.

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 $(S_j, \ell_{0,j}(\cdot)) = \texttt{feasible}(\mathcal{P}_j, \varepsilon, B_{0,j})$ 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 $i \geq 1$,

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

In the first inequality above we used the fact that, $\ell'_i(e) = \lceil \frac{B'_i}{B_i}\ell_i(e)\rceil \geq \frac{B'_i}{B_i}\ell_i(e)$ for $\succeq_i = \geq$, and that $\ell'_i(e) = \lfloor \frac{B'_i}{B_i}\ell_i(e)\rfloor \leq \frac{B'_i}{B_i}\ell_i(e)$ otherwise. In the second inequality above we used the fact that OPT is feasible w.r.t. the original problem. Moreover,

$$\begin{aligned}\ell'_0(OPT) &= \sum_{e\in OPT} \ell'_0(e) = \sum_{e\in OPT} \left\lfloor \frac{B'_0}{B_0}\ell_0(e)\right\rfloor \\ &\geq \sum_{e\in OPT} \frac{B'_0}{B_0}\ell_0(e) - 1 \geq B'_0(1+\varepsilon) - m \geq B'_0.\end{aligned}$$

In the second inequality above we used the assumption $\ell_0(OPT) \geq B_{0,j}(1+\varepsilon) = B_0(1+\varepsilon)$. In the third inequality above we used the fact that $B'_0 = \left\lceil \frac{m}{\varepsilon}\right\rceil \geq \frac{m}{\varepsilon}$.

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

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

In the second inequality above we used the feasibility of S_j and in the last inequality the assumption $\ell_0(OPT) < B_{0,j}(1+\varepsilon)^2$. The cost of $S^* = S_h$ then satisfies

$$\begin{aligned}\ell_0(S_h) &= \frac{B_{0,h}}{B'_{0,h}} \sum_{e\in S_h} \frac{B'_{0,h}}{B_{0,h}}\ell_0(e) \geq \frac{B_{0,h}}{B'_{0,h}} \sum_{e\in S_h} \left\lfloor \frac{B'_{0,h}}{B_{0,h}}\ell_0(e)\right\rfloor \\ &= \frac{B_{0,h}}{B'_{0,h}} \sum_{e\in S_h} \ell'_{0,h}(e) = \frac{B_{0,h}}{B'_{0,h}}\ell'_{0,h}(S_h) > \frac{\ell_0(OPT)}{(1+\varepsilon)^2}.\end{aligned}$$

The case $\succeq_0 = \leq$ is analogous. Here $B_{0,1} > \ldots > B_{0,q}$ are powers of $(1+\varepsilon)$, and we let j be the largest index such that $\ell_0(OPT) \leq B_{0,j}/(1+\varepsilon)$ (hence $\ell_0(OPT) > B_{0,j+1}/(1+\varepsilon) = B_{0,j}/(1+\varepsilon)^2$. Also in this case OPT is feasible for $\mathcal{P}'_j$. For $i \geq 1$, one can show that $\ell'_i(OPT) \succeq_i B'_i$ is the same way as in the previous case. Moreover

$$\begin{aligned}\ell'_0(OPT) &= \sum_{e\in OPT} \ell'_0(e) = \sum_{e\in OPT} \left\lceil \frac{B'_0}{B_0}\ell_0(e)\right\rceil \\ &\leq \sum_{e\in OPT} \frac{B'_0}{B_0}\ell_0(e) + 1 \leq \frac{B'_0}{1+\varepsilon} + m \leq B'_0.\end{aligned}$$

In the last inequality above we used the fact that $B'_0 = \left\lceil \frac{m(1+\varepsilon)}{\varepsilon}\right\rceil$. It follows that

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

We can conclude that that the cost of the approximate solution $S^* = S_h$ satisfies

$$\ell_0(S_h) = \frac{B_{0,h}}{B'_{0,h}} \sum_{e \in S_h} \frac{B'_{0,h}}{B_{0,h}} \ell_0(e) \leq \frac{B_{0,h}}{B'_{0,h}} \sum_{e \in S_h} \left\lceil \frac{B'_{0,h}}{B_{0,h}} \ell_0(e) \right\rceil = \frac{B_{0,h}}{B'_{0,h}} \sum_{e \in S_h} \ell'_{0,h}(e)$$
$$= \frac{B_{0,h}}{B'_{0,h}} \ell'_{0,h}(S_h) < \ell_0(OPT)(1+\varepsilon)^2. \qquad \square$$

We next prove the bitonicity of `feasible`: via the Composition Theorem 2.4 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. Consider an execution of $(S_j, \ell_{0,j}(\cdot)) =$ `feasible`$(\mathcal{P}_j, \varepsilon, B_{0,j})$ 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.

For two solutions $\bar{S}$ and S and two length vectors $\bar{\Lambda}$ and Λ, recall that by definition $\bar{\Lambda}(\bar{S}) \succeq \Lambda(S)$ iff, for some $j \in \{0, \ldots, k\}$, $\bar{\ell}'_i(\bar{S}) = \ell'_i(S)$ for $i = 0, \ldots, j$, and $\bar{\ell}'_{j+1}(\bar{S}) \succ_{j+1} \ell'_{j+1}(S)$ if $j < k$. 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 the 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. $\square$

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

Proof. It is sufficient to consider the case that the solution to the original problem is not $\emptyset$ nor $\mathcal{N}$. Consider the variant `ideal` of `multi` which spans all the (infinitely many) powers of $(1+\varepsilon)$. 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 $B_0 > m\ell_{0,max}\lceil \frac{m(1+\varepsilon)}{\varepsilon} \rceil$, 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 unfeasible. For $B_0 > m\ell_{0,max}\lceil \frac{m(1+\varepsilon)}{\varepsilon} \rceil$, 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 2.4, `ideal` is monotone. We can conclude that `multi` is monotone as well. □

The deterministic part of Theorem 1.1 follows from Lemmas 2.2, 3.1, and 3.3. The randomized part of Theorem 1.1 follows analogously by observing that one can make the failure probability of each execution of $\mathcal{A}$ (and hence of the overall algorithm) polynomially small.

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. We are also able to derandomize our PTAS in the special case of positive integer lengths.

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 [35])

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

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, using, say, Megiddo's parametric search technique (see, e.g., [35]).

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. The slope of $c_\lambda(S)$ is positive if S is unfeasible, 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. We will refer to $c_\lambda(S)$ as the *line* associated to S. When no confusion is possible, we will sometimes use the notion of spanning tree and of the corresponding line interchangeably. We observe the following useful fact.

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

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

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 $c(e)(1 + \varepsilon \frac{t_e}{2T^4})$

B2: Let $c_1 \leq \ldots \leq c_q$ be the powers of $(1+\varepsilon)$ between $c_{min}/(1+\varepsilon)$ and $c_{max}(1+\varepsilon)$. Let $1, \ldots, h$ denote all the pairs $(F, g(\cdot))$ with $F \subseteq E$, $|F| = \frac{1}{\varepsilon}$, 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: (i) contracting the edges of F_j; (ii) deleting the edges of value larger than $C_j := \min_{e \in F_j}\{g_j(e)\}$; (iii) decreasing L by $\ell(F_j)$. Compute $S_j =$ `lagrangian`$(\mathcal{P}_j)$.

B3: Return the solution $F_j \cup S_j$, $S_j \neq \mathcal{N}$, minimizing $c(F_j) + c(S_j)$, and maximizing j in case of ties ($\mathcal{N}$ if no such solution exists).

`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^-.

FIG. 3. *Algorithm* `bmst`.

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

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

□

4.1. Algorithm. Algorithm `bmst` is described in Figure 3. Let $\varepsilon \in (0, 1]$ be a given constant parameter. W.l.o.g., we can assume that the graph contains at least $2 + 1/\varepsilon$ nodes (and hence any spanning tree at least $1 + 1/\varepsilon$ edges). Otherwise, we can solve the problem optimally in polynomial time by brute force. Furthermore, the brute force algorithm can be easily made monotone with a careful implementation. The algorithm initially randomly perturbs the edge 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 two lines corresponding to spanning trees intersect at any 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 $1/\varepsilon$ edges, and $g_j : F_j \to C$, where $C = \{c_1, \ldots, c_q\}$ is a proper set of powers of $(1 + \varepsilon)$. Given a pair $j = (F_j, g_j(\cdot))$, subproblem $\mathcal{P}_j$ is obtained by contracting edges F_j, deleting all the edges of cost larger than $C_j := \min_{e \in F_j}\{g_j(e)\}$, and decreasing L by $\ell(F)$. Note that, when we contract an edge, this might create parallel edges. However this does not create any problem since `lagrangian` works for multi-graphs as well.

Each subproblem $\mathcal{P}_j$ is solved by means of a procedure `lagrangian`, based on the Lagrangian relaxation method. In particular, following [35], 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).

4.2. Analysis. 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 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 and S_2, 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, S_2 and S_3. With the same notation as above, these three lines intersect at the same point if and only if

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

for some value $\lambda \geq 0$. 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.

The following lemma bounds the approximation guarantee and efficiency of algorithm `bmst`.

LEMMA 4.3. *For any fixed $\varepsilon \in (0, 1]$, the expected running time of algorithm* `bmst` *is polynomial.*

Proof. The perturbation can be performed in $O(m \log(2T^4)) = O(m^2)$ time. Algorithm `bmst` runs at most $O((m \log_{1+\varepsilon} \frac{c_{max}}{c_{min}})^{1/\varepsilon})$ instances of `lagrangian`, which is

polynomial in the input size for any constant $\varepsilon > 0$. The running time of `lagrangian` is dominated, modulo polynomial factors, by the number of spanning trees of optimal Lagrangian cost for the considered instance. 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. □

LEMMA 4.4. *For any fixed $\varepsilon \in (0, 1]$, algorithm* `bmst` *is $(1 + 5\varepsilon)$-approximate.*

Proof. The algorithm can only return feasible solutions. Let us assume that a feasible solution exists, otherwise there is nothing to prove. Let OPT be the optimum solution to the perturbed instance, and F the $1/\varepsilon$ most expensive edges in OPT. Consider the assignment $g : F \to \{c_1, \ldots, c_q\}$ such that, for each $e \in F$, $g(e) \geq c(e) \geq g(e)/(1 + \varepsilon)$. 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 $\varepsilon(1 + \varepsilon)c(OPT)$. If $\lambda^* = 0$, $c(S^-) = c(OPT')$. Otherwise

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

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

It remains to prove that `bmst` is monotone. To that aim, we start by proving the bitonicity of the subroutine `lagrangian`.

LEMMA 4.5. *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, since the line of minimum-slope output in the original problem (cf. Step **L2**), will have minimum slope also in the modified instance. 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 4.6. *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 4.5, $f \in \bar{S}_j$ 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 4.5, $\bar{c}(\bar{S}_j) \geq c(S_j)$ which implies $\bar{c}(F_j \cup \bar{S}_j) \geq c(F_j \cup S_j)$. □

Lemma 4.7. *Algorithm* `bmst` *is monotone.*

Proof. It is sufficient to consider the case that the solution returned in the original problem is not $\mathcal{N}$. Analogously to the proof of Lemma 3.3, we define a variant `ideal` of `bmst` which considers all the powers of $(1+\varepsilon)$ 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 unfeasible. 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 $g_j(e) > c_{max}$ with the largest power of $(1+\varepsilon)$ not larger than $(1+\varepsilon)c_{max}$. Algorithm `ideal` is monotone by the Composition Theorem 2.4 and Corollary 4.6. It follows that `bmst` is monotone as well. □

The randomized part of Theorem 1.2 follows from Lemmas 2.2, 4.3, 4.4, and 4.7.

4.3. Derandomization for Integer Lengths. We next show how to derandomize the algorithm from Section 4.1 under the assumption that lengths are positive integers. W.l.o.g., we can assume that $L \geq 1$ is integer. By the same argument as before, we can assume that n is bounded from below by a sufficiently large constant. The basic idea is to exploit sufficiently small *additive* deterministic perturbations of the costs, so that an analogue of Lemma 4.2 holds. In order to guarantee that such perturbations are small enough, yet independent from agents' declarations, we round the costs in order to make them integral.

`bmst'`$(\mathcal{P}, \varepsilon)$

B1: Remove all edges e with $\ell(e) > L$.

B2: Let $c_1 \leq \ldots \leq c_q$ be the powers of $(1+\varepsilon)$ between $c_{min}/(1+\varepsilon)$ and $\frac{n(1+\varepsilon)}{\varepsilon} c_{max}$. Let $1, \ldots, h$ denote all the pairs $(F, g(\cdot))$ with $F \subseteq E$, $|F| = \frac{1}{\varepsilon}$, 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: (i) contracting the edges of F_j; (ii) deleting all the edges of cost larger than $C_j := \min_{e \in F_j}\{g_j(e)\}$; (iii) decreasing L by $\ell(F_j)$; (iv) replacing each cost $c(e)$ with $\lceil c(e)\frac{n}{\varepsilon C_j} \rceil + \delta(e)$. Compute $S_j :=$ `lagrangian`$(\mathcal{P}_j)$.

B3: Return the solution $F_j \cup S_j$, $S_j \neq \mathcal{N}$, minimizing $c(F_j) + c(S_j)$ and maximizing j in case of ties ($\mathcal{N}$ if no such solution exists).

FIG. 4. *Algorithm* `bmst'`. *Here* $\delta(e) := \frac{1}{M^{i(e)}}$, $i(e) \in \{1, \ldots, |E|\}$ *is the index of* e, *and* M *is a sufficiently large function of* ε, n *and* L.

In more detail, our deterministic variant `bmst'` of `bmst` is described in Figure 4. Initially edges with length larger than L, which do not belong to any feasible solution, are filtered out. This guarantees that, for any $\emptyset \neq E' \subseteq E$, $1 \leq \ell(E') \leq (n-1)L < nL$.

The construction of each problem $\mathcal{P}_j$ is analogous to the same construction in `bmst`, with the following differences. First of all, we update costs in a drastically different way. We initially replace each $c(e)$ with the rounded cost $\lceil c(e)\frac{n}{\varepsilon C_j} \rceil$, where $C_j := \min_{e \in F_j}\{g_j(e)\}$ is an upper bound on edge costs after Steps (i) and (ii). Observe that, after this rounding, edge costs are integers between 1 and n/ε. Then we add to each cost $c(e)$ the positive perturbation $\delta(e) := \frac{1}{M^{i(e)}}$. Here $i(e) \in \{1, \ldots, |E|\}$ denotes the position of edge e in an arbitrary, deterministic ordering of the edges of the graph. Furthermore, M denotes a sufficiently large function of n and L, to be fixed later. The second, technical difference is that we expand the set of candidates costs c_i w.r.t. the randomized variant: intuitively, we want to span all the values of C_j which induce distinct values of the rounded costs (before the perturbation). We remark that the (deterministic) procedure `lagrangian` is the same as in `bmst`.

Trivially, the overall algorithm is deterministic. Let us next prove that it has also the other desired properties. We first prove the following technical lemma about the properties of the perturbations. For a subset of edges F, let $\delta(F) := \sum_{e \in F} \delta(e)$. We also let $c'(F)$ be the modified cost of F and $c^r(F) = c'(F) - \delta(F)$ be the rounded (integral) cost of F before the perturbation.

LEMMA 4.8. *For any two distinct subsets of edges* F' *and* F'', *let* $f \in (F' \setminus F'') \cup (F'' \setminus F')$ *be the edge maximizing* $\delta(f)$ *(i.e., with the smallest index* $i(f)$*). Then*

$$\frac{M-2}{M-1}\delta(f) < |\delta(F') - \delta(F'')| < \frac{M}{M-1}\delta(f).$$

Proof. Observe that $|\delta(F') - \delta(F'')| = |\delta(F' \setminus F'') - \delta(F'' \setminus F')|$. Let $F_f := F' \cup F'' - \{f\}$.Trivially,

$$\delta(f) - \delta(F_f) \leq |\delta(F') - \delta(F'')| \leq \delta(f) + \delta(F_f).$$

The claim follows since $\delta(F_f) = \sum_{e \in F_f} \frac{1}{M^{i(e)}} < \delta(f) \sum_{i \geq 1} \frac{1}{M^i} = \delta(f)\frac{1}{M-1}$. ∎

LEMMA 4.9. *For any problem* $\mathcal{P}_j$, *all the spanning trees correspond to different lines, and at most two lines intersect at a given value of* $\lambda' \geq 0$.

Proof. Consider any two distinct spanning trees with edge sets E_1 and E_2. Assume by contradiction that the corresponding lines overlap, and hence $c'(E_1) = c'(E_2)$.

Then $|c^r(E_1) - c^r(E_2)| = |\delta(E_2) - \delta(E_1)|$. The latter equality cannot hold since the left-hand side is either 0 or at least 1, while the right-hand side is strictly between 0 and 1 by Lemma 4.8 (assuming $M > 2$).

Consider now any 3 distinct spanning trees, with edge sets E_1, E_2, and E_3, respectively. Define $c'_i := c'(E_i)$, for $i = 1, 2, 3$, and define similarly c^r_i, δ_i, and ℓ_i. Assume by contradiction that the corresponding lines intersect at a given value of $\lambda' > 0$. These lines do not overlap by the above argument, and cannot be parallel since they intersect in λ'. Then $c'_1 \neq c'_2 \neq c'_3 \neq c_1$. W.l.o.g. assume $c'_2 < c'_1 < c'_3$. Analogously to the proof of Lemma 4.2, one must have $\lambda' = \frac{\ell_1 - \ell_3}{\ell_2 - \ell_1} > 0$ and

$$c'_3 - c'_1 = \lambda'(c'_1 - c'_2) \Rightarrow |(c^r_3 - c^r_1) - \lambda'(c^r_1 - c^r_2)| = |\lambda'(\delta_1 - \delta_2) + \delta_1 - \delta_3|.$$

Observe that $\frac{1}{nL} < \lambda' < nL$ by the integrality of the lengths, and recall that the rounded costs c^r_i are integers between 1 and n/ε. Consequently $|(c^r_3 - c^r_1) - \lambda'(c^r_1 - c^r_2)|$ is either 0 or at least $\frac{1}{nL}$. We next prove that, for a large enough M,

$$0 < |\lambda'(\delta_1 - \delta_2) + \delta_1 - \delta_3| < \frac{1}{nL},$$

which gives the desired contradiction. For the upper bound part, let $f' \in (E_1 \setminus E_2) \cup (E_2 \setminus E_1)$ be the edge maximizing $\delta(f')$. Define f'' analogously w.r.t. edges in $(E_1 \setminus E_3) \cup (E_3 \setminus E_1)$. By Lemma 4.8, assuming $M > nL(nL+1)+1$, one obtains

$$\begin{aligned} |\lambda'(\delta_1 - \delta_2) + \delta_1 - \delta_3| &\leq |\lambda'(\delta_1 - \delta_2)| + |\delta_1 - \delta_3| \leq nL\frac{M}{M-1}\delta(f') + \frac{M}{M-1}\delta(f'') \\ &\leq \frac{nL+1}{M-1} < \frac{1}{nL}. \end{aligned}$$

For the lower bound part, assume by contradiction that

$$\lambda'(\delta_1 - \delta_2) + \delta_1 - \delta_3 = 0 \Leftrightarrow \lambda' = \frac{\delta_3 - \delta_1}{\delta_1 - \delta_2}.$$

Since $\lambda' > 0$, it must be the case that $\delta_3 > \delta_1 > \delta_2$ or $\delta_2 > \delta_1 > \delta_3$. Consider the first case, the second one being symmetric. By Lemma 4.8, with $M > 2$,

$$\frac{\delta_3 - \delta_1}{\delta_1 - \delta_2} = \frac{|\delta_3 - \delta_1|}{|\delta_1 - \delta_2|} \geq \left(\frac{M-2}{M-1}\delta(f'')\right) \cdot \left(\frac{M}{M-1}\delta(f')\right)^{-1} = \frac{M-2}{M} \cdot \frac{\delta(f'')}{\delta(f')} > 0. \qquad \square$$

From the previous proof, choosing $M = \Theta((nL)^2)$ is sufficient. Therefore perturbations can be encoded with a polynomial number of bits in the input size.

LEMMA 4.10. *For any fixed $\varepsilon \in (0, 1]$, algorithm* `bmst'` *runs in polynomial time.*

Proof. The proof follows along the same line as the proof of Lemma 4.3, the main difference being that in this case, in each execution of `lagrangian`, the number of trees of optimal Lagrangian cost is at most 2 deterministically due to Lemma 4.9. Thus each such execution, hence the overall algorithm, takes polynomial time. Here we also exploit the fact that $O(\log(M^{i(e)}))$ is polynomially bounded. $\square$

LEMMA 4.11. *For any $\varepsilon \in (0, 1]$, algorithm* `bmst'` *returns a $(1+8\varepsilon)$-approximate solution.*

Proof. If there is no feasible solution, there is nothing to prove. Otherwise, let OPT be the optimum solution, and F the $1/\varepsilon$ most costly edges in OPT. Consider the assignment $g : F \to \{c_1, \ldots, c_q\}$ such that, for each $e \in F$, $g(e) \geq c(e) \geq g(e)/(1+\varepsilon)$,

and the subproblem $\mathcal{P}_j$ induced by $j = (F, g(\cdot))$. Let OPT' be the optimum solution to $\mathcal{P}_j$, and OPT_c be the optimum solution to the same problem where one uses the original costs $c(\cdot)$ rather than the modified ones $c'(\cdot)$. By the same argument as in the proof of Lemma 4.4, $c(OPT_c) \leq c(OPT) - c(F)$. Furthermore, for $M \geq \frac{n+1}{n}$,

$$c'(OPT') \leq c'(OPT_c) \leq \frac{n}{\varepsilon C_j} c(OPT_c) + (n-1) + \sum_{i \geq 1} \frac{1}{M^i} \leq \frac{n}{\varepsilon C_j} c(OPT_c) + 2n.$$

By the same argument as in the proof of Lemma 4.4, Algorithm `lagrangian` returns a solution S^- such that $c'(S^-) \leq c'(OPT') + c'(e^-) \leq c'(OPT') + \frac{n}{\varepsilon} + \frac{1}{M}$. Since $c(S^-) \leq \frac{\varepsilon C_j}{n} c'(S^-)$, we can conclude that

$$\begin{aligned} c(S^-) &\leq \frac{\varepsilon C_j}{n} c'(S^-) \leq \frac{\varepsilon C_j}{n} (c'(OPT') + \frac{n}{\varepsilon} + \frac{1}{M}) \\ &\leq \frac{\varepsilon C_j}{n} (\frac{n}{\varepsilon C_j} c(OPT_c) + \frac{4n}{\varepsilon}) \leq c(OPT) - c(F) + 4C_j. \end{aligned}$$

The cost of the returned solution is therefore at most $c(F) + c(S^-) \leq c(OPT) + 4C_j \leq (1 + 8\varepsilon) c(OPT)$, where we used the fact that $C_j \leq \varepsilon g(F) \leq (1+\varepsilon)\varepsilon c(F) \leq (1+\varepsilon)\varepsilon c(OPT) \leq 2\varepsilon c(OPT)$. □

LEMMA 4.12. *Algorithm* `bmst'` *is monotone.*

Proof. We start by observing that the initial filtering Step B1 preserves the monotonicity of the rest of the algorithm: indeed, if f belongs to the final solution this means that $\ell(f) \leq L$ and consequently f is not filtered out in the modified problem where by assumption $\overline{\ell}(f) \leq \ell(f)$. If is sufficient to consider the case that the solution returned in the original problem is not $\mathcal{N}$.

Lemma 4.5 holds unchanged. Consider any edge f with $c(f) \neq \overline{c}(f)$, that is present in problems $\mathcal{P}_j$ and $\overline{\mathcal{P}}_j$. Let $c'(f)$ and $\overline{c}'(f)$ be the modified cost of f in the two problems. Then either $c'(f) = \overline{c}'(f)$ or the sign of $c(f) - \overline{c}(f)$ and $c'(f) - \overline{c}'(f)$ is the same. We can therefore conclude along the same lines as in the proof of Corollary 4.6 that the analogue of that corollary still holds: namely, the procedure which returns $F_j \cup S_j$ for a given problem $\mathcal{P}_j$ is bitonic w.r.t. $c(\cdot)$. Analogously to the proof of Lemma 3.3, we define a variant `ideal` of `bmst'` which considers all the powers of $(1+\varepsilon)$ for any guess. Also in this case the solution computed by `ideal` and `bmst'` is the same. In fact, when $C_j := min_{e \in F_j}\{g_j(e)\} < c_{min}$, all the edges are removed and the problem becomes unfeasible. Vice versa, for $C_j \geq \frac{n}{\varepsilon} c_{max}$, all the edge costs, before the perturbation, are rounded to 1. Hence, by rounding each $g_j(e) > \frac{n}{\varepsilon} c_{max}$ down to the largest power of $(1+\varepsilon)$ which is not larger than $\frac{(1+\varepsilon)n}{\varepsilon} c_{max}$, one obtains exactly the same solution: the modified assignment is considered by the original algorithm in some subproblem j'. Algorithm `ideal` is monotone by the Composition Theorem 2.4 and by the modified version of Corollary 4.6. It follows that `bmst'` is monotone as well. □

Combining Lemmas 2.2, 4.10,4.11, and 4.12, one obtains the deterministic part of Theorem 1.2.

5. PTAS for Independence Systems. In this section we consider the case that feasible solutions induce an independence system. Recall that in these problems removing any element from a feasible solution preserves feasibility. In particular, this means that $\mathcal{S}$ is closed under inclusion, i.e., $S \in \mathcal{S}$ implies $S' \in \mathcal{S}$ for any $S' \subseteq S$. Moreover, it must be $\succeq_i = \leq$ for all $i \geq 1$. The case *best* = min is trivial for

`ismulti`$(\mathcal{P}, \varepsilon)$

M1: Let $c_1 \leq \ldots \leq c_p$ be the powers of $(1+\varepsilon)$ between $\ell_{0,min}/(1+\varepsilon)$ and $\ell_{0,max}(1+\varepsilon)$. Let $B_{0,1} \geq \ldots \geq B_{0,q}$ be the powers of $(1+\varepsilon)$ between $\ell_{0,min}\frac{1}{m(1+\varepsilon)}$ and $m\,\ell_{0,max}\lceil\frac{m(1+\varepsilon)}{\varepsilon}\rceil$. Let $1,\ldots,h$ denote all the triples $(F, g(\cdot), B_{0,r})$ with $F \subseteq \mathcal{U}$, $|F| \leq \frac{k}{\varepsilon}$, $g : F \to \{c_1,\ldots,c_p\}$ and $1 \leq r \leq q$. For a given triple $j = (F_j, g_j(\cdot), B_{0,r_j})$, define subproblem $\mathcal{P}_j$ by: (i) selecting the elements of F_j; (ii) removing all the elements e not in F_j with $\ell_0(e) > \min_{e' \in F_j}\{\ell_0(e')\}$ (all the edges if $|F| < \frac{k}{\varepsilon}$); (iii) decreasing B_i by $\ell_i(F_j) = \sum_{e\in F_j}\ell_i(e)$ for $i \geq 1$. Compute $(S_j, \ell_{0,j}) =$ `isfeasible`$(\mathcal{P}_j, \varepsilon, B_{0,r_j})$.

M2: Return the solution $F_j \cup S_j$, $S_j \neq \mathcal{N}$, maximizing $\ell_0(F_j) + \ell_{0,j}(S_j)$, and maximizing j in case of ties ($\emptyset$ if no such solution exists).

`isfeasible`$(\mathcal{P}, \varepsilon, B_0)$

F1: Let $B'_0 = \lceil\frac{m}{\varepsilon}\rceil$ and $I_0 = \{B'_0, B'_0+1, \ldots, m\,B'_0\}$. For all $e \in \mathcal{U}$, replace $\ell_0(e)$ with $\min\{\ell_0(e), B_0\}$ and define $\ell'_0(e) = \lfloor\frac{B'_0}{B_0}\ell_0(e)\rfloor$.

F2: For $i = 1,\ldots,k$, let $B'_i = m^2$ and $I_i = \{0, 1, \ldots, B'_i\}$. Discard all the elements with $\ell_i(e) > B_i$, and define $\ell'_i(e) = \lceil\frac{B'_i}{B_i}\ell_i(e)\rceil$ for the remaining elements.

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 the largest incidence vector in lexicographic sense 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 $(S^*_z, \frac{B_0}{B'_0}\ell'_0(\cdot))$.

FIG. 5. *Algorithm* `ismulti`.

these problems, since the optimum solution is the empty one. Hence we also assume *best* = max. Altogether, these problems can be formulated as

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

Also in this case we focus on the case that the exact version of the problem admits a deterministic pseudo-polynomial-time algorithm $\mathcal{A}$: the randomized case is analogous. We make the same assumptions as in Section 3. Furthermore, we implicitly assume that we can select an element e in the sense described in the introduction.

Our algorithm `ismulti` is described in Figure 5. The basic idea is to combine the approach of `multi` with the novel guessing technique of `bmst`. The algorithm generates (Step M1) 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. Moreover, as in `multi`, in each subproblem we also set up a lower bound on the cost of the desired solution.

More precisely, each subproblem $\mathcal{P}_j$ is labeled by a triple $(F_j, g_j(\cdot), B_{0,r_j})$. In this subproblem the elements of F_j are selected, and each budget B_i, $i \geq 1$, is decreased

by $\ell_i(F_j)$. Function $g_j(\cdot)$ assigns a rounded cost to the elements of F_j similarly to `bmst`. All the elements of cost larger than $\min_{e\in F_j}\{\ell_0(e)\}$ not in F_j are discarded. Each subproblem $\mathcal{P}_j$ is solved by means of a procedure `isfeasible`, inspired by `feasible`. The unique difference is that the modified lengths $\ell'_i(e)$, for $i \geq 1$, are equal to $\lceil \frac{\ell_i(e)}{B_i} B'_i \rceil$, where $B'_i = m^2$. Procedure `isfeasible` returns a solution S_j and a cost function $\ell_{0,j}(\cdot)$. This procedure is designed in order to be bitonic with respect to $\ell_{0,j}(\cdot)$ on subproblem $\mathcal{P}_j$.

Eventually `ismulti` returns the (feasible) solution S_j that maximizes $\ell_0(F_j) + \ell_{0,j}(S_j)$, the best solution with largest index j in case of ties (Step M2).

LEMMA 5.1. *For any fixed $\varepsilon \in (0,1]$,* `ismulti` *runs in polynomial time and it returns a feasible solution which is $\frac{1}{(1+\varepsilon)^4}$-approximate.*

Proof. The number of subproblems generated by `ismulti` is $O(m^{k/\varepsilon}(\log^2_{1+\varepsilon} \frac{m\ell_{0,max}}{\varepsilon \ell_{0,min}}))$, which is polynomial being k constant. Similarly to Algorithm `multi`, we notice that `isfeasible` takes time $O((m/\varepsilon)^{O(1)})$, thus implying that the algorithm takes polynomial time overall.

Assume that OPT contains at least k/ε elements, the remaining case being simpler. Let F_j be the k/ε most expensive elements in OPT, and $OPT' = OPT \setminus F_j$. Let us restrict our attention to the triples of kind $(F_j, g_j(\cdot), B_{0,r_j})$, where $g_j(e) \leq \ell_0(e) \leq (1+\varepsilon) g_j(e)$ for all $e \in F_j$. We denote by e_i the element of largest length ℓ_i in OPT', $i \geq 1$. We next show that $\ell'_i(OPT' \setminus \{e_i\}) \leq B'_i$. Let $|OPT'|$ denote the number of elements in OPT'. Observe that $\ell_i(OPT' \setminus \{e_i\}) \leq \frac{|OPT'|-1}{|OPT'|}\ell_i(OPT') \leq \frac{m-1}{m}\ell_i(OPT')$. Then

$$\begin{aligned}\ell'_i(OPT' \setminus \{e_i\}) = \sum_{e\in OPT'\setminus\{e_i\}} \ell'_i(e) &\leq |OPT'| - 1 + \sum_{e\in OPT'\setminus\{e_i\}} \frac{\ell_i(e)}{B_i} B'_i \\ &\leq m - 1 + \frac{m-1}{m}\frac{\ell_i(OPT')}{B_i} B'_i \leq m + \frac{m-1}{m} B'_i = B'_i.\end{aligned}$$

As a consequence $OPT'' := OPT' \setminus \{e_1, \ldots, e_k\}$ satisfies $\ell'_i(OPT'') \leq B'_i$ for all $i \geq 1$. Since the largest cost in the problems considered is at most $\min_{e\in F_j}\{\ell_0(e)\}$, we have that $\ell_0(OPT'') \geq \ell_0(OPT') - k \min_{e\in F_j}\{\ell_0(e)\}$. By the usual argument, OPT'' is a feasible solution to the rounded problem $\mathcal{P}'_j$ associated to the triple $(F_j, g_j(\cdot), B_{0,r_j})$, with $B_{0,r_j}(1+\varepsilon) \leq \ell_0(OPT') - k \min_{e\in F_j}\{\ell_0(e)\} < B_{0,r_j}(1+\varepsilon)^2$. Hence a solution S_j is returned for that problem. For this solution we have $\ell'_0(S_j) \geq B'_0$ which implies $\ell_0(S_j) \geq B_{0,r_j}$. We then have

$$\begin{aligned}\ell_0(S_j \cup F_j) =& \ell_0(OPT) - \ell_0(OPT') + \ell_0(S_j) \\ \geq& \ell_0(OPT) - \ell_0(OPT') + B_{0,r_j} \\ \geq& \ell_0(OPT) - \ell_0(OPT') + \frac{1}{(1+\varepsilon)^2}(\ell_0(OPT') - k \min_{e\in F_j}\{\ell_0(e)\}) \\ \geq& \ell_0(OPT) - \ell_0(OPT') + \frac{1}{(1+\varepsilon)^2}(\ell_0(OPT') - \varepsilon\,(\ell_0(OPT) - \ell_0(OPT'))) \\ =& (1 - \frac{\varepsilon}{(1+\varepsilon)^2})\ell_0(OPT) - (1 - \frac{1}{(1+\varepsilon)^2} - \frac{\varepsilon}{(1+\varepsilon)^2})\ell_0(OPT') \\ \geq& (1 - \frac{\varepsilon}{(1+\varepsilon)^2})\ell_0(OPT) - (1 - \frac{1}{(1+\varepsilon)^2} - \frac{\varepsilon}{(1+\varepsilon)^2})\ell_0(OPT) \\ \geq& \frac{1}{(1+\varepsilon)^2}\ell_0(OPT).\end{aligned}$$

Let $S_d \cup F_d$ be the solution returned by the algorithm. Similarly to the proof of Lemma 3.1, we can observe that $\ell_{0,j}(S_j) \geq \ell_0(S_j)/(1+\varepsilon)^2$. We can conclude that

$$\begin{aligned}\ell_0(S_d \cup F_d) &\geq \ell_0(F_d) + \ell_{0,d}(S_d) \geq \ell_0(F_j) + \ell_{0,j}(S_j) \\ &\geq \frac{1}{(1+\varepsilon)^2}\ell_0(F_j \cup S_j) \geq \frac{1}{(1+\varepsilon)^4}\ell_0(OPT). \qquad \square\end{aligned}$$

LEMMA 5.2. *Algorithm* `ismulti` *is monotone.*

Proof. It is sufficient to consider the case that the solution to the original problem is not $\emptyset$. By an argument analogous to the proof of Lemma 3.2, `isfeasible` is bitonic on problem $\mathcal{P}_j$ with respect to the cost function $\ell_{0,j}(\cdot)$. By the same argument as in Corollary 4.6, the procedure which returns $F_j \cup S_j$ for a given problem $\mathcal{P}_j$ is bitonic with respect to the cost function used in step M2 of `ismulti`. Analogously to the proof of Lemmas 3.3 and 4.7, we define a variant `ideal` of `ismulti` which considers all the powers of $(1+\varepsilon)$ for any guess. Also in this case it is easy to see that the solution computed by `ideal` and `ismulti` is the same. By the Composition Theorem 2.4 and the bitonicity of the procedure above, we can conclude that `ideal` is monotone. It follows that `ismulti` is monotone as well. $\square$

Theorem 1.3 follows from Lemmas 2.2, 5.1, and 5.2. The randomized part of Theorem 1.3 follows along the same line as in the case of `multi`.

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 *TARK*, pages 72–87, 2003.

[4] A. Berger, V. Bonifaci, F. Grandoni, and G. Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle, *Mathematical Programming* 128(1-2): 355–372, 2011.

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

[6] V. Bilu, V. Goyal, R. Ravi, and M. Singh. On the crossing spanning tree problem. In *APPROX-RANDOM*, pages 51–64, 2004.

[7] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. *SIAM J. Computing*, 40(6): 1587–1622, 2011.

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

[9] C. Chekuri, J. Vondrák, and R. Zenklusen, Multi-budgeted matchings and matroid intersection via dependent rounding. In *SODA*, pages 1080–1097, 2011.

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

[11] J. R. Correa and A. Levin, Monotone Covering Problems with an Additional Covering Constraint, *Mathematics of Operations Research*, 34(1):238–248, 2009.

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

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

[14] A. Fiat, A. Goldberg, J. Hartline, and A. Karlin. Competitive generalized auctions. In *STOC*, pages 72–81, 2002.

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

[16] F. Grandoni, P. Krysta, S. Leonardi, and C. Ventre. Utilitarian Mechanism Design for Multi-Objective Optimization. In *SODA*, pages 573-584, 2010.

[17] F. Grandoni, R. Ravi, and M. Singh. Iterative rounding for multi-objective optimization problems. In *ESA*, pages 95–106, 2009.

[18] F. Grandoni, R. Ravi, M. Singh, and R. Zenklusen. New Approaches to Multi-Objective Optimization. To appear in *Mathematical Programming*.

[19] F. Grandoni, R. Zenklusen. Approximation schemes for multi-budgeted independence systems. In *ESA*, pages 536–548, 2010.
[20] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.
[21] R. Hassin. Approximation schemes for the restricted shortest path problem, *Mathematics of Operation Research* 17(1): 36–42, 1992.
[22] 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.
[23] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *FOCS*, pages 595–604, 2005.
[24] D. J. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49(5):577–602, 2002.
[25] D. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest paths problem, *Operations Research Letters* 28: 213–219, 2001.
[26] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III. Bicriteria network design problems. In *ICALP*, pages 487–498, 1995.
[27] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *AAAI*, pages 379–384, 2002. Full version in *Games and Economic Behavior*, 64(2), pages 612–631, 2008.
[28] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. In *STOC*, pages 345–354, 1987.
[29] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.
[30] N. Nisan and A. Ronen. Algorithmic mechanism design. In *STOC*, pages 129–140, 1999.
[31] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.
[32] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *FOCS*, pages 86–92, 2000.
[33] P. Penna and C. Ventre. Optimal Collusion-Resistant Mechanisms with Verification. *Games and Economic Behavior*, 2012. In press. DOI: j.geb.2012.09.002.
[34] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *FOCS*, pages 202–213, 1994.
[35] R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem. In *SWAT*, pages 66–75. 1996.
[36] 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 *STOC*, pages 438–447, 1993.
[37] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.
[38] 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.
[39] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *J. Finance*, 16:8–37, 1961.
[40] A. Warburton. Approximation of Pareto optima in multiple-objective, shortest path problems. *Operations Research*, 35: 70–79, 1987.