OR Spectrum (2020) 42:965–994
https://doi.org/10.1007/s00291-020-00597-7

REGULAR ARTICLE

Strategyproof auction mechanisms for network procurement

Martin Bichler[1] · Zhen Hao[1] · Richard Littmann[1,2] · Stefan Waldherr[1]

Received: 13 March 2018 / Accepted: 10 July 2020 / Published online: 25 July 2020

Abstract

Deferred-acceptance auctions can be seen as heuristic algorithms to solve $\mathcal{NP}$-hard allocation problems. Such auctions have been used in the context of the Incentive Auction by the US Federal Communications Commission in 2017, and they have remarkable incentive properties. Besides being strategyproof, they also prevent collusion among participants. Unfortunately, the worst-case approximation ratio of these algorithms is very low in general, but it was observed that they lead to near-optimal solutions in experiments on the specific allocation problem of the Incentive Auction. In this work, which is inspired by the telecommunications industry, we focus on a strategic version of the minimum Steiner tree problem, where the edges are owned by bidders with private costs. We design several deferred-acceptance auctions (DAAs) and compare their performance to the Vickrey–Clarke–Groves (VCG) mechanism as well as several other approximation mechanisms. We observe that, even for medium-sized inputs, the VCG mechanisms experiences impractical runtimes and that the DAAs match the approximation ratios of even the best strategy-proof mechanisms in the average case. We thus provide another example of an important practical mechanism design problem, where empirics suggest that carefully designed deferred-acceptance auctions with their superior incentive properties need not come at a cost in terms of allocative efficiency. Our experiments provide insights into the trade-off between solution quality and runtime and into the additional premium to be paid in DAAs to gain weak group-strategyproofness rather than just strategyproofness.

Keywords Steiner tree problem · Network procurement · Approximation mechanism · Deferred-acceptance auction

✉ Martin Bichler
martin.bichler@in.tum.de

Extended author information available on the last page of the article

1 Introduction

There is a significant literature in the design of approximation algorithms for computationally hard problems (Vazirani 2013). Algorithmic mechanism design extends this literature in an important way (Nisan and Ronen 1999). The goal of approximation mechanisms is the design of computationally efficient algorithms which take into account the incentives of participants as well. These mechanisms should run in polynomial time and satisfy strong game-theoretical equilibrium solution concepts such that bidders have incentives to reveal their valuations truthfully and the auctioneer can determine the optimal allocation or one that approximates the optimal solution. This has led to a rich literature studying approximation mechanisms for different types of $\mathcal{NP}$-hard resource allocation problems. Typically, designers of approximation mechanisms aim for dominant-strategy incentive-compatibility or strategyproofness. Such mechanisms are prior-free, and truthful bidding is a dominant strategy for individual bidders.

Network procurement is a prime application where auction mechanisms play an important role in business practice. A telecom is interested in connecting several sites or terminals via a cost-minimal set of edges connecting vertices in a network. The terminals constitute a subset of all vertices in the network, and suppliers can provide individual edges in the network at a certain cost. The minimum Steiner tree problem is a well-known model of this network procurement problem, and even with complete information about suppliers' costs, finding a cost-minimal solution is $\mathcal{NP}$-hard. The minimum Steiner tree problem on graphs is one of the most well-known $\mathcal{NP}$-complete problems (Karp 1972), and central in various types of network design problems, which have received significant attention in operations research (Xu et al. 1995; Öncan et al. 2008; Contreras and Fernández 2012).

In the procurement environment, the cost of establishing a link is the private information of its supplier. Each supplier wants to maximize her payoff, i.e. her bids minus their private costs for setting up the connection. In such an auction, the auctioneer wants to set incentives for bidders to reveal their costs truthfully. It is well known that the Vickrey–Clarke–Groves (VCG) (Vickrey 1961; Clarke 1971; Groves 1973) mechanism is the only quasi-linear mechanism which maximizes social welfare and is strategyproof (Green and Laffont 1977). Still, the resulting discounts can be manipulable by coalitions of suppliers, a property which can well be a problem in procurement. This means the VCG mechanism is not group-strategyproof. In addition, the VCG mechanism is no longer strategyproof if the allocation does not maximize social welfare, i.e. if the allocation cannot be solved exactly. Since the minimum Steiner tree problem is $\mathcal{NP}$-complete, its optimal solution, which corresponds to the maximally achievable social welfare, cannot be expected to be obtained in reasonable time.

If the allocation cannot be computed optimally, but only approximately, then the VCG mechanism loses this strong game-theoretical property (Lehmann et al. 2002). This paper analyzes several well-known approximation algorithms for the minimum Steiner tree problem with respect to their implementability in settings where the edges of the graph are strategic agents. Based on well-known theory from mechanism design, we verify that some of these approximation algorithms can be extended to strategyproof mechanisms, while others are not.

Motivated by the Incentive Auction of the US Federal Communications Commission (FCC), Milgrom and Segal (2019) and Leyton-Brown et al. (2017) recently proposed *deferred-acceptance auctions (DAAs)*, a class of greedy algorithms which are weakly group-strategyproof for bidders with single-dimensional types. This means even a coalition of bidders cannot manipulate profitably via deviations from truthful bidding, which makes them robust against collusive bidding strategies. This is a very desirable property in many applications. Also, a deferred-acceptance auction can be implemented both as a sealed-bid and as a clock auction.

An important question is whether these strong incentive properties are at the expense of solution quality, i.e. they might lead to low allocative efficiency. Dütting et al. (2017) derived worst-case approximation ratios for two important problem classes. Still, for most problems no worst-case approximation ratios have been proven. Interestingly, experimental analysis of the specific allocation problem in the US FCC Incentive Auction showed very high solution quality on average (Newman et al. 2017). In their simulations, which focused on the efficiency of the reverse auction, the reverse clock auction achieved highly efficient solutions. The specific scoring rule by the FCC played an important role in the solution quality and the payments computed. The allocation problem in the Incentive Auction is special, and it is not clear whether one could achieve high average efficiency with a DAA also for other problems.

We perform a thorough computational study in which we compare DAA variants to more sophisticated approximation mechanisms for the Steiner minimum tree problem. The results show that in general, the DAA (with an adequately chosen scoring function) results in high solution quality, but that in environments with a very sparse network and few terminals, primal-dual algorithms or Mehlhorn's algorithm is better. All approximation algorithms and heuristics were computed within only two minutes on average, while the computation times for exact solutions with a Vickrey–Clarke–Groves payment rule are extensive and took more than 18 hours on average for the larger instances. The revenue is lowest in the Vickrey–Clarke–Groves mechanism. The DAA variants led to higher payments for the buyer, which can be seen as a premium paid for group-strategyproofness, i.e. its robustness to collusion. Our empirical results illustrate the order of magnitude of these trade-offs.

In Sect. 2, we introduce related literature, before we introduce the minimum Steiner tree and relevant definitions in Sect. 3. In Sect. 4, we analyze the implementability of well-known approximation algorithms for the minimum Steiner tree problem, and a critical payment scheme, before we introduce deferred-acceptance auctions. Then, in Sect. 6 the results of numerical experiments based on the SteinLib are presented.

2 Related literature

The minimum Steiner tree problem has many important applications in a variety of fields. Examples include biology (phylogenetic trees), the design of integrated circuits, and it occurs as a special case or subproblem in many other problems in the field of network design (single-sink rent-or-buy, prize-collecting Steiner tree, single-sink buy-at-bulk). Due to its relevance, the problem received a lot of attention and different classes of algorithms emerged.

Approximation algorithms based on distance networks were proposed by Takahashi and Matsuyama (1980) and Kou et al. (1981). Mehlhorn (1988) developed a faster variant of the latter algorithm. All algorithms in this class achieve an approximation ratio of 2, which is also achievable by means of primal-dual algorithms, see e.g. Goemans and Williamson (1995). Loss-contracting approximations are another class of algorithms studied in the context of the minimum Steiner tree problem. This approach has been improved in a series of papers. The algorithm due to Robins and Zelikovsky (2005) currently reaches the best approximation ratio of 1.55. Byrka et al. (2010) proposed a randomized technique that achieves an approximation ratio of $ln(4) + \epsilon$, i.e. 1.39 in the limit. While the algorithm can be derandomized to obtain a deterministic approximation algorithm with polynomial time complexity, the polynomial and constants required to reach the approximation factor of 1.39 result in a runtime which is not feasible in practice. In our analysis, we start with the best known approximation algorithm by Robins and Zelikovsky (2005), before we analyze the approach by Mehlhorn (1988), and primal-dual algorithms (Goemans and Williamson 1995). These are arguably the most prominent approaches to the minimum Steiner tree problem in the literature.

We focus on the design of approximation mechanisms, i.e. approximation algorithms that can be implemented in dominant strategies. The field of algorithmic mechanism design has made substantial progress in the past years, and there are general frameworks to achieve truthfulness with randomized approximation mechanisms, and deterministic approximation mechanisms for specific problems. For example, a well-known black-box method to convert approximation algorithms for any packing problem into strategyproof mechanisms is the framework by Lavi and Swamy (2011), which is a randomized approximation algorithm.

Yet randomized approximation algorithms are often not acceptable in industrial procurement. Unfortunately, as of now there is no general framework to transform deterministic approximation algorithms into strategyproof mechanisms. However, there exist quite general approaches when additional conditions on bidders' valuations are met. Single-mindedness has received most attention in the literature on combinatorial auctions (Lehmann et al. 2002). It means that bidders are only interested in one specific subset of items (package). This can be a reasonable assumption for many real-world markets, and it is a very good starting point for our analysis of strategyproof approximation mechanism for the minimum Steiner tree problem on graphs. In the context of network procurement, we talk about bidders with single-dimensional types, which means each supplier only having access to a single link which she can sell.

Mu'alem and Nisan (2008) extended the framework of Lehmann et al. (2002) and presented conditions for approximately efficient and strategyproof mechanisms and single-minded bidders. Apart from this, numerous approximation mechanisms have been developed for specific algorithmic problems such as parallel scheduling and maximum flow problems (Archer and Tardos 2001), or graph traversal problems (Bilò et al. 2007). Interestingly, in spite of the importance of the minimum

Steiner tree problem, it has received very little attention in the literature on algorithmic mechanism design so far, with the only prior work being due to Gualà and Proietti (2005). They present a distance-network-based approximation mechanism which draws on the ideas of Takahashi and Matsuyama (1980).

We are particularly interested in the new class of deferred-acceptance auctions, which were introduced by Milgrom and Segal (2019) in the context of the Incentive Auction design for the US Federal Communications Commission (Leyton-Brown et al. 2017). Little is known so far about the solution quality deferred-acceptance auctions as compared to other deterministic and strategyproof approximation mechanisms in general. Dütting et al. (2017) is an exception, and they discuss approximation ratios of deferred-acceptance auctions for knapsack auctions as well as general combinatorial auctions with single-minded bidders. Recently, deferred-acceptance auctions have been generalized by Gkatzelis et al. (2017) for non-binary settings in which bidders do not simply win or lose but receive some level of service (e.g. a number of items awarded in a multi-item auction).

3 Notation and definitions

Let $G = (V, E, c)$ be a weighted, connected graph, where c_e is the cost of each edge $e \in E$. For a subset of edges $F \subseteq E$, the cost of the edge-induced subgraph is defined by $c(F) = \sum_{e \in F} c_e$. A *spanning tree* of G is a subset of edges of E such that the resulting edge-induced subgraph is connected, cycle-free and contains all vertices V. The *minimum spanning tree*, denoted by $MST(G)$, is a spanning tree where the sum of the costs of its edges is minimal in comparison with all other spanning trees.

The minimum Steiner tree problem on a connected graph $G = (V, E, c)$ is defined as follows. For a subset of vertices $K \subseteq V$ called *terminals*, any tree spanning K is called a *Steiner tree*. Any vertex in a Steiner tree which is not a terminal is called a *Steiner point*. We refer to the set of all Steiner trees over G as $StT(V, E)$. The objective then is to find a minimum cost Steiner tree.

Let G_V be the complete graph induced by the vertex set V, i.e., a complete weighted graph $G_V = (V, E_V, c_V)$, where each edge cost equals the cost of the shortest path in G between the two adjacent vertices of that edge. G_V is then a metric graph satisfying the triangle inequality. We call G_V the *distance network* of the graph G. Likewise, G_K denotes the distance network induced by the terminal set K, $G_K = (K, E_K, c_K)$. Note that $G_K \subseteq G_V$, as $K \subseteq V$.

In the following, we describe the design of mechanisms for the minimum Steiner tree problem. We consider a set of bidders N, where bidders $i \in N$ have single-dimensional types, i.e. each bidder i only provides one specific single edge e_i. With slight abuse of notation, we denote with c_i the true cost of bidder i while c refers to the corresponding tuple $(c_i)_{i \in N}$ taken over all bidders. Denote with B_i the domain of bids, i can report as her cost for edge e_i, e.g. $B_i = R_{\geq 0}$. B is defined as the Cartesian product $\prod_{i \in N} B_i$. For a *single-dimensional* bidder i, there is a unique and publicly known edge $e_i \in E$ such that her true private cost is c_i only for edge e_i, while for all other edges $e_j \neq e_i$ her true private cost is ∞. Given a vector of reported bids $b \in B$ with $b = (b_i)$, the expression b_{-i} denotes the bid tuple without the i-th entry, $b_{-i} = (b_j)_{j \in E \setminus \{i\}}$, and (c_i, b_{-i}) denotes the bid tuple where the i-th entry of b is replaced by c_i, i.e., bidder i reports her true cost.

A deterministic mechanism $\mathcal{M} = (f, p)$ for the minimum Steiner tree problem over vertices V and edges E is defined by a deterministic allocation function $f : B \to StT(V, E)$ and a payment scheme $p_i : B \times StT(V, E) \to \mathbb{R}$ for each bidder i. Given the bidders' reported bids $b \in C$, the mechanism $\mathcal{M} = (f, p)$ computes a Steiner tree $f(b)$ and pays each bidder i a payment of $p_i(b, f(b))$. In an approximation mechanism, the allocation function f is implemented via a deterministic approximation allocation algorithm $\mathcal{A}$. A mechanism with an approximation allocation algorithm $\mathcal{A}$ achieves an approximation ratio of r for minimum Steiner tree if

$$\max_{b \in B} \frac{c(OPT(b))}{c(\mathcal{A}(b))} \leq r$$

where $OPT(b)$ denotes a welfare-maximizing allocation (i.e. an optimal minimum Steiner tree given costs b), $c(OPT(b))$ the corresponding social welfare (i.e. cost of the Steiner tree), and $c(\mathcal{A}(b))$ the welfare achieved with the approximation algorithm $\mathcal{A}$.

Since bidders are self-interested, their reported bids b do not necessarily reflect their true costs c. Instead, bidders try to maximize their quasi-linear utilities u_i, i.e., payment received minus true cost: $u_i(b) = p_i(b, f(b)) - c_i$. As a result, a strategy-proof mechanism must offer bidders some incentives to reveal their true costs.

Definition 1 (*Strategyproofness*) A mechanism $\mathcal{M} = (f, p)$ is strategyproof if for all bidders $i \in E$ and all reported bid tuples $b \in B$ it holds that bidder i has a weakly higher payoff by telling the truth:

$$u_i(c_i, b_{-i}) \geq u_i(b)$$

Then, a bidder cannot make herself better off by not telling the truth about her costs. We also consider the stronger criterion of weak group-strategyproofness, where groups of bidders cannot make themselves better off by colluding.

Definition 2 (*Weak Group-Strategyproofness*) A mechanism $\mathcal{M} = (f, p)$ is weakly group-strategyproof if for every set of bidders $I \subseteq E$ and all reported bid tuples $b \in B$ it holds that at least one bidder $i \in E$ has a weakly higher payoff by telling the truth:

$$u_i(c_I, b_{-I}) \geq u_i(b)$$

In other words, in a weakly group-strategyproof mechanism it is impossible for a group of bidders to find alternative (non-truthful) bids that make all members of the group strictly better off.

We assume w.l.o.g. that for any two bidders i, j with $i \neq j$, it is $e_i \neq e_j$. If there are multiple bidders providing the same edge, we only consider the lowest reported bid for the allocation algorithm (though, of course, we consider all bids for the payment scheme). So from now on, we assume $i \widehat{=} e_i$. To avoid monopoly, we restrict G to be 2-edge-connected, i.e., G remains connected even if any single edge is removed.

With this, we can now formulate the minimum Steiner tree problem as a mechanism design problem: Let $G = (V, E, b)$ be a 2-edge-connected graph. $|V|$ is the number of vertices, $|E|$ is the number of edges/bidders, and b is the vector of reported bid prices. Let $K \subseteq V$ be the set of terminals. Then, the objective is to design a polynomial time approximation mechanism which computes an approximately efficient allocation A, and a payment scheme p which makes truthful bidding a dominant strategy, such that p and A form a strategyproof mechanism.

Definition 3 (*Monotonic allocation rule*) An allocation rule f of a mechanism $\mathcal{M} = (f, p)$ is monotonic if a bidder i who wins with bid b_i keeps winning for any lower bid $b'_i < b_i$ (for any fixed settings of the other bids).

Definition 4 (*Critical payment scheme*) A payment scheme p of a mechanism $\mathcal{M} = (f, p)$ is critical if a winning bidder i receives payment p^*_i, which is her maximum bid allowed for winning: $p^*_i := \sup\{b'_i \in B_i : i \in A(b'_i, b_{-i})\}$, where $A(b'_i, b_{-i})$ denotes the set of bidders that would have won if the reported bids were (b'_i, b_{-i})

Intuitively, a monotonic allocation ensures that a winner remains winning with any better bid, while the critical payment for a winning bidder is the highest cost that she may declare and still win. In his seminal paper, Myerson (1981) showed that an allocation rule f is implementable (i.e., there exists a payment vector p such that $\mathcal{M} = (f, p)$ is strategyproof) if and only if the allocation rule is monotonic. Moreover, if the allocation rule is monotonic and losing bidders pay 0, a critical payment scheme is the unique payment rule p such that $\mathcal{M} = (f, p)$ is strategyproof. Hence, with single-dimensional types and monotonic approximation algorithms, we can implement an outcome in dominant strategies, if we compute critical payments.

4 Approximation mechanisms for single-dimensional bidders

In this section, we briefly introduce important approximation algorithms for the minimum Steiner tree problem. A more extensive discussion can be found in the appendix. For the algorithms which can be extended to approximation mechanisms, we provide a corresponding critical payment scheme in Sect. 4.2, which then yields a strategyproof approximation mechanism. Finally, in Sect. 5 we design a deferred-acceptance auction for the minimum Steiner tree problem and discuss the worst-case approximation ratio of the deferred-acceptance auction and general greedy algorithms.

4.1 Approximation algorithms for the Steiner minimum tree

Table 1 lists the approximation algorithms for the minimum Steiner tree that we compare to the greedy approximation algorithms used in the deferred-acceptance auctions. These algorithms are representatives of very different approaches to

Table 1 Selected approximation algorithms for the minimum Steiner tree problem on graphs

Year	Approx. ratio	Authors	Monotonic?	Paradigm
1988	2.00	Mehlhorn	Yes	Distance network
1997	2.00	Goemans, Williamson	Yes	Primal-dual
2005	1.55	Robins, Zelikovsky	No	Loss-contracting

approximation. While two of them are monotonic, the one by Robins and Zelikovsky (2005) with the best approximation ratio so far is not.

We only provide an overview with a classification of whether these algorithms are monotonic or not. While this is fairly straightforward to answer for most approximation algorithms in the literature, the currently best class of algorithms for the minimum Steiner tree problem, the loss-contraction algorithms, are challenging to analyze. In the appendix we provide a detailed description of these algorithms and proofs of their monotonicity.

4.1.1 Loss-contracting algorithms

Loss-contracting algorithms have been the most successful approach to the design of approximation algorithms for the minimum Steiner tree on graphs so far. Any Steiner tree $S(G, K)$ of G is either a full Steiner tree, i.e., all its terminals are leaves, or can be decomposed into a forest of full Steiner subtrees (full components) by splitting all the non-leaf terminals (splitting a terminal results in two copies of the same terminal). The algorithm by Robins and Zelikovsky (2005) builds an MST on the subgraph G_K induced by the terminal set K and repeatedly adds full components to improve the temporary solution. In each iteration, full components are ranked according to their gain (by how much the component improves the current temporary solution) divided by their loss (i.e., the cost committed by adding a component or more precisely its Steiner points). After a full component is added, the temporary solution is improved. This step also involves loss-contracting, a method to make components which are in conflict with added ones less appealing. By these means, the algorithm by Robins and Zelikovsky (2005) achieves an approximation ratio of 1.55 if $k \rightarrow \infty$ and it is computable in $\mathcal{O}(|K|^k \cdot |V - K|^{k-2} + k \cdot |K|^{2k+1} \log |K|)$. This is the best approximation algorithm so far, but unfortunately it is not monotonic.

Proposition 1 *Allocation algorithm A^{RZ} is not monotonic.*

4.1.2 Distance-network-based approximations

Similarly to the loss-contracting approximation, the general idea of distance-network-based approximation algorithms is to build a MST on a complete subgraph $\overline{G_K}$ in the first phase. In the second phase, edges in $MST(\overline{G})$ are re-transformed back to edges in G, and an MST is computed on the resulting graph to remove possible cycles. Finally, in the third phase, non-terminal leaves are deleted. This algorithm was proposed by Kou et al and runs in $\mathcal{O}(|K||V|^2)$. However, due to the cycles that can occur in the first phase, this standard variant is not monotonic. Mehlhorn (1988) designed an algorithm which differs in phase 1. Here, the algorithm first partitions G into Voronoi regions, which are then utilized to construct a subgraph of G_K, called $\overline{G}$. It then proceeds with phase 2 and phase 3. This leads to a worst case runtime of $\mathcal{O}(|V| \log |V| + |E|)$ and achieves an approximation ratio of $2(1 - 1/l)$ where l is the minimal number of leaves in any minimum Steiner tree (which is naturally bounded above by the number of terminals). This algorithm is monotonic.

Proposition 2 *The allocation algorithm A^{MH} by Mehlhorn is monotonic.*

4.1.3 Primal-dual approximation algorithms

The approximation algorithm for the minimum Steiner tree problem by Goemans and Williamson (1995) follows a primal-dual approach in which the minimum Steiner tree problem is transformed into a hitting set problem and modeled as an integer linear program (IP). By relaxing the IP and considering its dual, Goemans and Williamson (1995) where able to propose an approximation algorithm that requires a runtime of $O(|V|^2 \log |V|)$ and also has an approximation ratio of 2.

Proposition 3 *The allocation of the primal-dual-based* minimum Steiner tree *approximation A^{PD} is monotonic.*

4.2 Payment schemes

Since the allocations of the algorithms A^{MH} by Mehlhorn (1988) and the primal-dual algorithm A^{PD} by Goemans and Williamson (1995) are monotonic, they meet the first requirement to be extendable to a strategyproof approximation mechanism. For the second requirement, the payment scheme p needs to find the critical payment p_i^* for any winner i, such that every reported bid b_i with $b_i \leq p_i^*$ is guaranteed to win, while every reported bid b_i with $b_i > b_i^*$ is guaranteed to lose.

We first discuss a payment scheme for distance-network-based approximation algorithms initially introduced by Gualà and Proietti (2005) which can be computed in $\mathcal{O}((|V| + |K|^2)|E| \cdot log\ \alpha(|E|, |V|))$. For this, consider the basic structure of an algorithm based on the distance network. Each winning edge e lies on at least one path connecting two terminals (v_1, v_2). If we now increase the cost of e, there are two possible causes that lead to e getting excluded from the solution. Either, there might be a shorter path between v_1 and v_2 that e is not part of. Apart from this, even if e is still on the shortest path between v_1 and v_2, the edge (v_1, v_2) might be replaced by some other edge (v_1', v_2') in the MST of distance network for which e is not on the resulting shortest path. The critical payment for e is then calculated by adding the difference between the original cost of the shortest path including e and the minimum cost of one of these alternatives without e.

The corresponding values can be calculated similar to (Gualà and Proietti 2005): First, the all-to-all distances problem is solved. Then, for every winning edge e on a shortest path between terminals v_1, v_2, an alternative shortest path between v_1 and v_2 that does not contain e can efficiently be computed using several tweaks (Buchsbaum et al. 1998; Gualà and Proietti 2005; Pettie and Ramachandran 2002). Computation of an alternative path in the distance network between two different terminals v'_1, v'_2 can also be done efficiently by standard sensitivity analysis (Tarjan 1982).

While the previous approach can be used for distance-network-based approximation, there is no efficient scheme for calculating the payments for the primal-dual approach and the loss-contraction algorithm by Robins and Zelikovsky (2005). In this case, another possibility to obtain critical payments is to find them through binary search. For a winning bidder i, a starting interval $[b_i, sp(b_{-i})]$, and first provisional payment $p^i_0 := \left\lfloor \frac{b_i + sp(b_{-i})}{2} \right\rfloor$, the binary search recursively computes a sequence of payments (p^i_j):

$$p^i_j = \begin{cases} \left\lfloor \frac{1}{2} p^i_{j-1} \right\rfloor & \text{if} \quad e_i \notin \mathcal{A}(G_{p^i_{j-1}}, K) \\ \left\lfloor \frac{3}{2} p^i_{j-1} \right\rfloor & \text{if} \quad e_i \in \mathcal{A}(G_{p^i_{j-1}}, K) \end{cases}$$

where $G_{p^i_{j-1}} = (V, E, (p^i_{j-1}, b_{-i}))$ is the original graph G with the only change being that the reported bid price of bidder i for his edge e_i has now become p^i_{j-1} instead of b_i, and $\mathcal{A}$ is the corresponding approximation algorithm. So $e_i \in \mathcal{A}(G_{p^i_{j-1}}, K)$ means that reporting p^i_{j-1}, bidder i still wins with all other bids fixed. This means that each computed payment p^i_j is tested by the respective allocation algorithm.

Since the payments described above are critical, they can be used in combination with their corresponding approximation algorithm to yield a strategy-proof approximation mechanism for single-dimensional bidders due to the results of Nisan et al. (2007).

5 Deferred-acceptance auctions

Greedy algorithms are an important class of approximation algorithms. A greedy-in algorithm iteratively chooses the best available option based on the current state (i.e., the previous iterations) and adds it to the solution. Contrarily, in the deferred-acceptance auction (DAA), a greedy-out algorithm is used which removes the least favorable alternative from the solution in every iteration. A greedy-in procedure which greedily accepts edges is not suitable for constructing a Steiner tree, since greedily accepting edges leads to being forced to 'correct' the structure afterwards (e.g. assuring that Steiner points do not end up as leaves), while within a greedy-out procedure one only needs to assure that it is still possible to construct a Steiner tree based on the remaining edges (i.e. all terminals are still connected). Thus, this section describes a greedy-out approximation for the Steiner tree problem implemented as a DAA (Milgrom and Segal 2019). The DAA is not only strategyproof but also weakly group-strategyproof and therefore provides a form of protection against bidder collusion.

The DAA greedily excludes the least desirable option from the solution until further removal would lead to an infeasible solution. To decide which option should be excluded in each iteration a scoring function is used. A scoring function assigns a value of at least 0 to an option i based on the cost of i and the remaining options. It is important to note that only the presence of remaining options, not their cost, may be considered in the scoring function as otherwise the mechanism might lose its incentive properties. Also, a scoring function needs to be non-decreasing in the first argument (cost of i). In each iteration, the option with the highest assigned score is removed from the allocation, options that cannot be removed without making the resulting solution infeasible receive a score of 0. All remaining edges with a score of 0 are accepted in the end. Hence, the algorithm always returns a feasible Steiner tree at the end. A representation of the algorithm is given below (Algorithm 1).

```
Data: 2-connected graph $G = (V, E, b)$, terminal set $K \subseteq V$, a scoring
      function $s$
Result: A Steiner tree in $G$ spanning $K$.
while true do
    for each edge $e$ do
        assign score $s(e)$ to $e$
        if $s(e) = 0$ then
            compute payment $p(e)$
        end
    end
    if highest score equals 0 then
        return remaining edges (Steiner tree)
    end
    remove $e$ with highest score
end
```

Algorithm 1: Deferred-acceptance auction (DAA)

The payment $p(e)$ for an alternative e is calculated the moment we cannot exclude e from the solution any more, i.e. the moment we assign a score of 0 to it. The payment is equal to the bid e could have stated such that her score would have been equal to the one removed in the last iteration.

In a network procurement context, the set of options is the set of edges E. An edge cannot be excluded from the solution if its removal would lead G to decay into two connected components each of which contains at least one terminal. To account for the specific requirements of the network procurement context, we analyze three scoring functions in our experimental analysis:

1. the weight of the edge

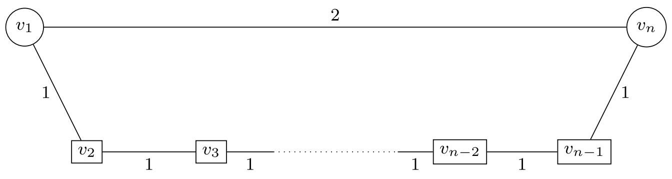


Fig. 1 An example where DAA leads to an approximation ratio of $(n-1)/2$

2. the weight of the edge divided by the number of adjacent edges
3. the weight of the edge divided by the betweenness centrality of the edge, where the betweenness centrality for each edge is defined by the number of shortest paths within the graph that use this edge

We calculate betweenness centrality by using the algorithm by Kourtellis et al. (Kourtellis et al. 2015) on a variant G_u of G where all edges have weight 1, i.e. G_u is the unweighted version of G. This is necessary, since due to incentive reasons a scoring function may only take the respective bid and the underlying graph structure into account, not the bids of other active bidders. Since in our environment bids are the cost of edges, when calculating the score of an edge e, we must ignore the costs of all other edges $e' \neq e$ in our calculations. In the following, let DAA_w (DAA_a;DAA_c) denote the DAA with scoring by weight (divided by adjacent edges; betweenness centrality).

Proposition 4 *The DAA for the* minimum Steiner tree *problem runs in* $\mathcal{O}(|E|^2 + |E||V| + t)$ *including payment calculation where t is the time necessary to update the scores.*

Proof In each iteration, at most $|E|$ scores need to be updated. This takes $|E| + |V|$ operations once to check for connectivity by Tarjan's bridge finding algorithm (Tarjan 1974) and constant time to update the score. Since there are at most $|E|$ iterations, this leads to a total runtime of $\mathcal{O}(|E|^2 + |E| \cdot (|E| + |V|)) \subseteq \mathcal{O}(|E|^2 + |E||V|)$ for DAA_w and DAA_a. For DAA_c betweenness centrality has to be calculated. This can be done in $\mathcal{O}(|E||V|)$ for each removal, i.e. in each iteration, using Kourtellis' algorithm (Kourtellis et al. 2015). Hence, the total runtime for DAA_c is $\mathcal{O}(|E|^2 + |E||V| + |E|^2|V|) \subseteq \mathcal{O}(|E|^2|V|)$. In all variants, payments need to be calculated for at most $|E|$ edges in constant time each. Thus, the runtime complexity is dominated by score updates. □

Greedy algorithms are usually fast, but can lead to arbitrarily bad results compared to an optimal solution for some problems. For instance, consider the three weight functions discussed above and a network as shown in Fig. 1. The network consists of n nodes $v_1, \ldots, v_n$, two of which (v_1 and v_n) are terminals. The optimal Steiner tree consists of only keeping edge (v_1, v_n), but under DAA this edge is rejected first under all weight functions, forcing the algorithm to accept all other $n-1$ edges in order to remain connected. This leads to an approximation ratio of

$(n-1)/2$ proving the impossibility of a constant-factor approximation ratio. It remains an open question whether there exists a weight function that allows for such an approximation ratio.

6 Experimental evaluation

In the following, we present a thorough analysis of the different mechanisms discussed in this paper. For the approximation mechanism based on Mehlhorn (1988) and the primal-dual algorithm of Goemans and Williamson (1995), we computed the payments as described in Sect. 4.2. For the former, we employed the payment scheme for distance-network-based approximation algorithms by Gualà Proietti, and for the latter, we calculated the payments based on binary search. For the DAA, we use the threshold payments which are dynamically updated throughout the run of the algorithm as described in Sect. 5. Finally, we also included the Vickrey–Clarke–Groves mechanism as a baseline. We used the send-and-split method (Erickson et al. 1987) as implemented by Iwata and Shigemura (2018) to determine optimal solutions. All algorithms were implemented in Java. The experiments were executed on a laptop with Intel core i5-6600k (4 cores, 3.5 GHz) and 8GB RAM. We first describe the data set in Sect. 6.1 before we presenting our results in Sect. 6.2.

6.1 Data

Experiments are conducted on set I080 of the SteinLib Testdata Library (Koch et al. 2000).[1] Instances which are not 2-edge-connected are not considered since a monopoly edge would be worth infinite amounts of money. Thus, instances with names ending on 0*x* or 3*x* are not included. The remaining instances covered graphs with 4, 8, 16, and 20 terminals and densities of 11, 20, and 100 percent (very sparse as well as complete graphs). For each combination of terminal and density, the SteinLib test set contained 5 instances, i.e. a total of 60 instances. In order to investigate the effect of a graph's density on the performance of algorithms, we created additional instances with more diverse density values. Based on complete instances in I080, we created instances with densities between 0.3 and 0.9 (in increments of 0.1) by deleting edges randomly. For each combination of number of terminals and density, we generated 5 instances, for a total of 140 additional instances. Overall, we have an extensive set of 200 instances and 6 algorithms, resulting in 1200 experiments.

6.2 Results

Let us now summarize the results with respect to allocative efficiency, runtime, and revenue. We conclude this section with a short discussion of the results, comparing the different mechanisms.

[1] http://steinlib.zib.de/showset.php?I080.

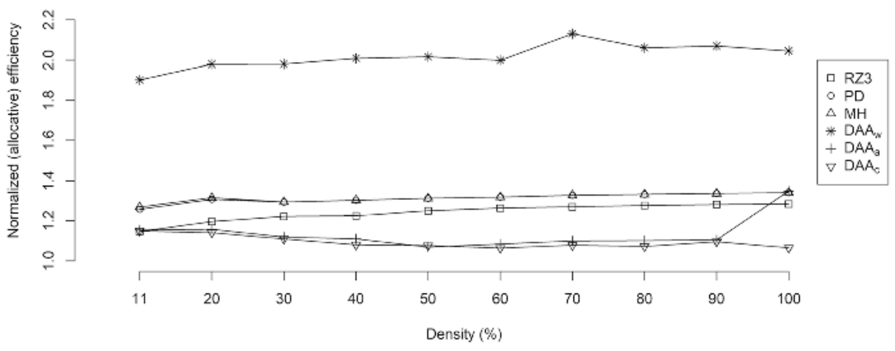


Fig. 2 Average efficiency over all instances

6.2.1 Allocative efficiency

In Fig. 2, we illustrate the efficiency of the five algorithms considered in our experimental evaluation for different levels of density. For each level of density, we show the mean efficiency of the algorithm for 20 instances (5 instances each for 6, 8, 16, and 20 terminals). While the approximation algorithm by Robins and Zelikovsky (2005) is not monotonic and thus cannot be extended to an approximation mechanism, it is still interesting to compare its allocation efficiency to the other algorithms in a complete information setting. Overall, DAA_c and DAA_a were the best performing algorithms and the scoring function based on the betweenness centrality came out to be the best scoring function for DAAs.

With a paired Wilcoxon rank-sum test, the differences in efficiency between *MH* and *PD* ($p = 0.059$) were not significant at $p < 0.0001$, while all other pairwise comparisons were significant at this level. We also analyze differences in efficiency using a linear regression with efficiency as dependent variable, the algorithm, the number of edges and terminals as covariates. With the DAA_c as baseline, the differences to this greedy algorithm were positive and significant at the following levels: DAA_w ($p < 0.0001$), *MH* ($p < 0.0001$), *PD* ($p < 0.0001$), *RZ* ($p < 0.0001$), and DAA_a ($p < 0.01$). The estimated coefficients further allow us to order the algorithm with respect to efficiency. Since we used the DAA_c as the baseline and all estimated coefficients are positive, we see that this approach provides the best results. The DAA_a (coefficient: 0.04) and the algorithm by Robins and Zelikovsky (coefficient: 0.14) follow closely, while both Mehlhorn's algorithm and the primal-dual approach exhibit a coefficient of 0.22. Finally, the DAA_w has the highest coefficient of 0.92.

Let us now report averages for different subgroups of the experiments in more detail. The algorithm by Robins and Zelikovsky performs best for sparse instances, on average finding a solution only 25% worse than the optimum and even solutions as good as 1.01 times the optimum (instance I080-015 of SteinLib). Moreover, it performs well for complete graphs, too (1.3 approximation ratio). Mehlhorn's algorithm and the primal-dual algorithm achieve similar results (130–140% of the optimum). The performance of these approximation algorithms gets slightly worse the denser the graph is.

The performance of the DAA heavily depends on the scoring function. Using only the weight of an edge c_e as a score, allocative efficiency is never better than 1.48 times of the optimum, usually worse than 1.6 times of the optimum. It seems clear that without taking into account the structure of the graph, the greedy algorithm employed in DAA_w cannot compete with more sophisticated methods. Even in later stages of the DAA, edges are only selected based on their individual cost without considering the possible paths this edge is a part of. DAA_c (and especially, for smaller densities, DAA_a) generally provides better results on average than the more sophisticated approximation mechanisms. Only on very sparse instances, the algorithm by Robins and Zelikovsky can keep up with the performance of the DAA variants DAA_a and DAA_c. If we use edge weight divided by number of adjacent edges as scoring function, DAA_a performs better than the primal-dual algorithm for most sets of instances and even achieves results that are better than the results of the algorithm by Robins and Zelikovsky, except for the instances with 100 percent density.

The performance of DAA_a decreases significantly between a 90 percent density and 100 percent density. We generated further instances, increasing the density by a single percentage point between 90 and 100 percent. The efficiency ratio steadily increases between 90 and 100 percent. While DAA_a is equivalent to DAA_w in the first stages in very dense graphs (since every edge has the same number of adjacent edges), this effect does not come into play except for very dense graphs. Overall, DAA_a performs significantly better than DAA_w (on a significance level of 0.1%). Moreover, it can be seen that efficiency of DAA_a and DAA_c is nearly identical for sparse graphs and even up to a density level of 90 percent. In sparse graph, the number of possible paths between two nodes is more limited. Since an edge e with a lot of adjacent edges naturally allows for more paths (and hence also more shortest paths) to pass through e, the betweenness centrality of e is very dependent on its adjacent edges. Therefore, the DAAs with the corresponding scoring functions perform very similarly.

Figure 3 shows the average performance of the algorithms depending on the number of terminals (averaged over all densities, i.e. 50 instances per number of terminals). It can be seen that performance of all DAA variants improves as the number of terminals grows. This is to be expected since a greedy-out procedure actually solves MST optimally and the Steiner minimum tree problem becomes more like the MST problem for an increasing number of terminals (in the limit, when all vertices are terminal, they are identical). In contrast, all other approximation algorithms perform worse the more terminals are present in the graph.

In the following, we discuss the efficiency of the algorithms for fixed number of terminals. Since results for 6 and 8 as well as for 16 and 20 terminals, respectively, are very similar, we consider instances with a small number (6) and large number (20) of terminals. To improve readability, in the following we excluded DAA_w from the graphs and discussion as it was performing worse than all other mechanisms, in general. For 6 terminals, the density appears not to have a significant impact for MH, the primal-dual approach, and DAA_a (Fig. 4). This can easily be seen by means of linear regression where the p-value of density is > 0.1 for all of them. The algorithm by Robins and Zelikovsky performs worse the denser the graph is ($p < 0.001$), an effect easily observable in Fig. 4. In contrast, the DAA_c

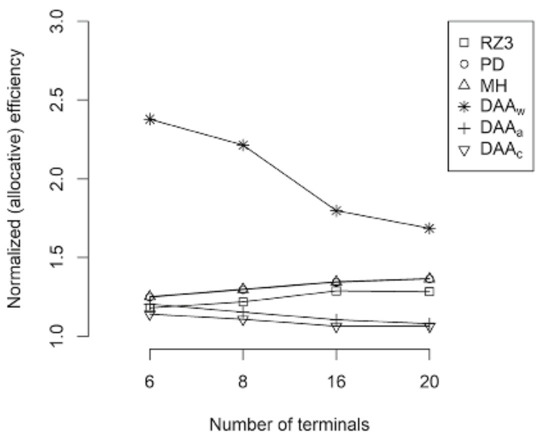


Fig. 3 Average efficiency depending on number of terminals

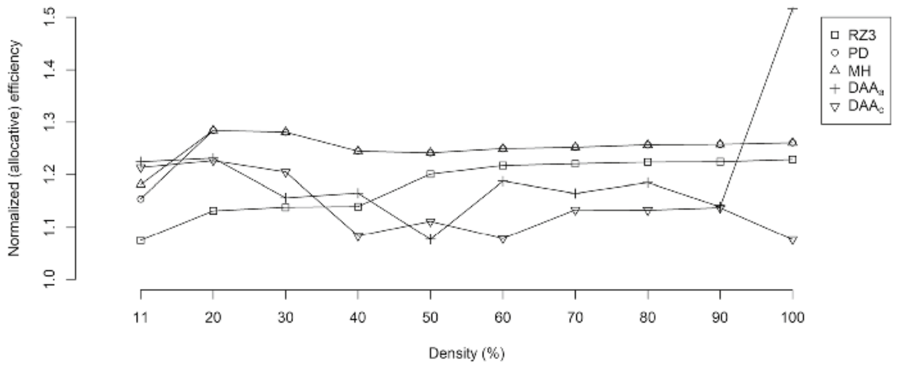


Fig. 4 Efficiency for 6 terminals

reveals favorable behavior if the graph is denser ($p < 0.01$). We can clearly see that independent of the number of terminals, DAA_a performs well for density levels up to 90 percent and then becomes significantly worse for very dense graphs.

This is in line with the results for 20 terminals where the density leads to better efficiency of the DAA_a for density values in [0.11, 0.9] ($p < 0.001$).

Figures 4 and 5 show that the *MH* and *PD* find very similar solutions that are below 1.4 times the optimal solution. The denser the graph, the worse the solutions these algorithms find. The algorithm by Robins and Zelikovsky finds even better solutions, but its performance decreases not only with increasing density but also as the number of terminals grows. Only for sparse instances with few terminals, it finds more efficient solutions than the more sophisticated DAA variants. DAA_a and the DAA_c find similarly efficient solutions for sparse graphs, where no significant difference was observed. The solutions found by the DAA_c are never worse than 1.09 times the optimum and in eight out of ten instances at most 6% worse than the optimal value for complete graphs with 16 or 20 terminals.

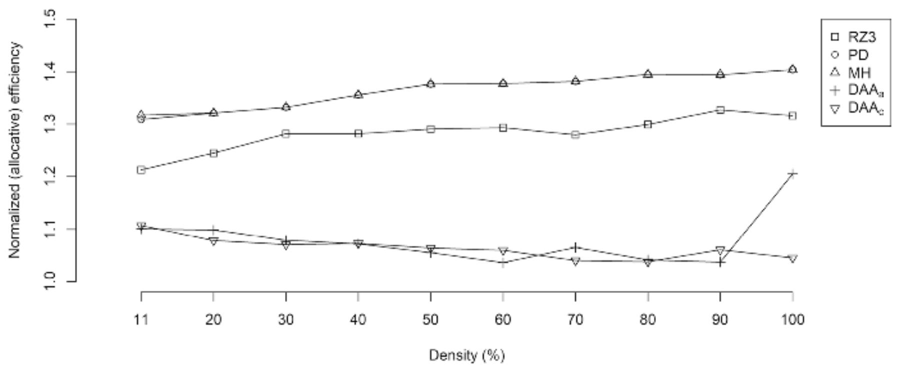


Fig. 5 Efficiency for 20 terminals

Table 2 Runtime combined means (in seconds)

Density	6 Terminals		20 Terminals	
	11%	100%	11%	100%
MH	0.24	0.46	0.33	1.26
PD	0.28	3.14	1.93	12.97
DAA_w	0.33	2.55	0.14	2.52
DAA_a	0.71	3.03	0.28	3.25
DAA_c	5.82	109.11	4.50	95.14
VCG	2.05	1.80	67544.54	58556.20

6.2.2 Runtime

In the following, we discuss the combined runtime required for the approximation mechanism to obtain both allocation and payments. Since the relative runtimes between mechanisms show a continuous pattern when incrementing the number of terminals and density, we only discuss extreme cases. Table 2 depicts the runtimes for the smallest and highest densities, as well as the smallest and largest numbers of terminals. Over all instances and densities, pairwise differences between two algorithms are significant at a level of $p < 0.0001$ using a Wilcoxon rank-sum test with the difference between the mechanism based on Mehlhorn's algorithm and the DAA_w being the only exception ($p = 0.026$). All mechanisms are significantly different from *VCG* ($p < 0.0001$). Our experiments show that *MH*, DAA_w, and DAA_A are the fastest group with *PD* following closely for instances with 6 or 8 terminals. Runtimes observed for the fastest group are lower than $4s$ on average on the set of instances with high numbers of both terminals and edges. Performance for lower numbers of terminals or edges is even better. Within the fastest group, the mechanism by Mehlhorn takes less time on complete graphs but is usually outperformed by the faster DAA variants for sparse graphs with more than 6 terminals.

Arguably, the more advanced payment computation used within *MH* leads to faster completion than calculating prices for *PD*, although we observed higher

Table 3 Revenue combined means (in arbitrary bid's currency)

Density	6 Terminals		20 Terminals	
	11%	100%	11%	100%
MH	2004.40	1496.00	6569.80	5580.00
PD	1920.40	1493.40	6654.80	5578.20
DAA_w	3376.60	2703.00	7393.60	6822.80
DAA_a	2152.27	2937.95	6133.26	9561.76
DAA_c	2735.85	2708.28	7404.98	6904.70
VCG	1644.80	1191.60	5078.80	4170.20

allocation runtime of the primal-dual algorithm when prices were not considered. The runtime required by all DAA variants is very dependent on the density of the graph, while it scales very well with the number of terminals. In our test instances, the computation time even slightly decreases with an increasing number of terminals in contrast to all other mechanisms. VCG in particular is very sensitive to higher number of terminals. Calculating exact solutions for minimum Steiner tree problems in the case of 20 terminals proved to be computationally inefficient, requiring a factor of up to 370.000 of the runtime as compared to the best performing DAA variants. Differences between the simpler scoring functions DAA_w and DAA_a are very small, while the calculation of betweenness centrality leads to higher runtimes for the DAA_c.

6.2.3 Revenue

Table 3 shows the extreme cases with the lowest density and highest density as well as the lowest and largest number of terminals. We have decided on this type of report, because again the developments between the extremes are smooth. In general, the payment increases with the number of terminals (since more edges need to be bought) and decreases with the density (since more competition between bidders allows for selecting cheaper options). The only exception is DAA_a with 100 percent density due to the bad (and hence more expensive) outcome that is obtained by the algorithm. Overall, the payments are lowest for VCG and the more sophisticated approximation algorithms yield a lower payment than the DAA variants (except for DAA_a for very sparse graphs).

The differences between DAA_w and DAA_c, *MH* and DAA_w, *MH* and DAA_c, *PD* and DAA_w, *PD* and DAA_c and *VCG* and all other algorithms are significant at a $p < 0.0001$ level using a Wilcoxon rank-sum test. The other pairs were not significantly different at this level, and their revenues are close. A regression with the sum of payments as the dependent variable and the number of edges, the number of terminals, and the algorithm as predictor variables shows that there are significant differences in revenue among most mechanisms. With *PD* as a baseline, we find that this approach yields lower payments than DAA_w ($p < 0.0001$), DAA_a ($p < 0.0001$), and DAA_a ($p < 0.0001$). No significant difference to the payments computed for the mechanism based on Mehlhorn's algorithm was found at this level. Using the *VCG*

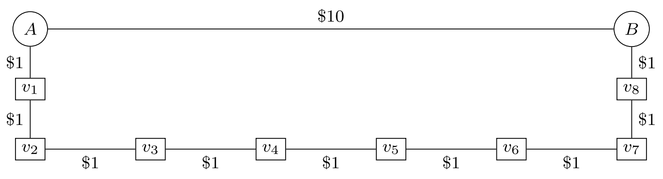


Fig. 6 An example where DAA leads to high payments

Table 4 Average seller utility (in % of their cost)

Density	6 Terminals		20 Terminals	
	11%	100%	11%	100%
MH	19.87	1.33	11.55	1.05
PD	15.15	1.25	12.64	1.02
DAA_w	2.65	1.69	2.71	1.45
DAA_a	22.46	70.41	22.01	106.54
DAA_c	46.22	103.66	48.08	65.70
VCG	13.65	1.85	12.51	6.00

mechanism as a baseline, all other algorithms yield significantly higher payments ($p < 0.0001$).

We also report results on the mean total utility calculated as the difference of total cost and total payments as percentage of total cost in Table 4. For *MH* and *PD*, the payments offered to winners of complete instances are low compared to their costs leading to low payoffs. For the DAA_a and DAA_c, sellers get a much higher payoff in all settings. The costs using the VCG mechanism are lowest.

Overall, we observe that the DAA mechanisms require significantly higher payments than VCG or the two approximation algorithms. While all of the mechanisms are strategy-proof, the allocation and payments are computed very differently. To see this, consider the example in Fig. 6 with two nodes *A* and *B* connected via a direct edge whose cost is $10. There is another path between these two nodes with 9 edges with a cost of $1 each. With threshold payments in a DAA, each of the winning edges gets $10 when the direct edge is removed, and the overall revenue of the bidders on the 9 winning edges is $90. With critical payments as they were used for MH and PD, the payments on each edge would be the maximum bid that would have still made the specific bidder winning (i.e., $2 - \varepsilon$), and the resulting revenue would be less than $18.

7 Conclusions

In this paper, we showed which approximation algorithms can be extended to approximation mechanisms for the Steiner minimum tree problem with single-minded bidders. We proved that the best known algorithm by Robins and Zelikovsky violates monotonicity, a necessary condition for implementability. However, the algorithms by Mehlhorn (1988) and the primal-dual algorithm by Goemans and Williamson (1995) are monotonic and could be extended to strategyproof approximation mechanisms. Further, we designed a deferred-acceptance auction for the minimum Steiner tree problem and analyzed several scoring functions.

While the worst-case approximation ratio of deferred-acceptance auctions can be very low, the average-case solution quality is remarkably high, as shown in our numerical experiments. The results show that the group-strategyproof DAA with a scoring function based on betweenness centrality (DAA_c) yields very high allocative efficiency at low computation times in most environments (characterized by density and the number of terminals). DAA_a with a scoring function based on the number of adjacent edges performs similarly well (and even better for graphs with a higher number of terminals). Only for very dense graphs, the algorithm's solution quality declines.

Obviously, in terms of allocative efficiency, the exact *VCG* mechanism is best, but runtime can be prohibitive as the number of terminals grows. The runtime of all other algorithms is less than two minutes even for the large instances. *MH* exhibits a very low runtime over all instances and lower payments than DAA variants. Its allocative efficiency is significantly worse than that of DAA_a and DAA_c except for very sparse graphs with a low number of terminals. Overall, DAA_a has the best solution quality for density levels between 20 and 90 percent.

A number of questions are left for future research. The group-strategyproofness and the high average solution quality of the DAA variants come at the cost of higher payments for the buyer and higher profit for the sellers. This interesting trade-off for procurement managers requires further analysis to better understand which conditions justify the additional payment.

Another extension is to consider cases where bidders not only provide single edges, but packages of multiple edges. Unfortunately, this extension towards multi-dimensional mechanism design is far from trivial. One of the main problems is that bidders might not only lie about the cost of their edges but also about which edges they possess. This considerably complicates the construction of truthful mechanisms, such that we cannot rely on critical payments and monotonicity any more. In general, the design of deterministic approximation mechanisms for hard computational problems with multi-minded bidders remains a challenge.

Acknowledgements Open Access funding provided by Projekt DEAL. This project was supported by the Deutsche Forschungsgemeinschaft (DFG) (BI 1057/1-8) and (GRK 2201).

Appendix: Approximation algorithms for the minimum Steiner tree

In Section "Loss-contracting approximation: the algorithm by Robins and Zelikovsky," we discuss loss-contraction algorithms on the basis of the algorithm by Robins and Zelikovsky (2005), in Section "Distance-network-based approximations," we consider distance-network-based approaches, and in Sect. 1, the approximation by a primal-dual algorithm.

Loss-contracting approximation: the algorithm by Robins and Zelikovsky

Any Steiner tree $S(G, K)$ of G is either a full Steiner tree, i.e., all its terminals are leaves, or can be decomposed into a forest of full Steiner subtrees (full components) by splitting all the non-leaf terminals (splitting a terminal results in two copies of the same terminal). The algorithm builds a *MST* on the subgraph G_K induced by the terminal set K and repeatedly adds full components to the temporary solution. In each iteration, full components are ranked according to their gain (by how much the component improves the current temporary solution) divided by their loss (i.e. the cost committed by accepting a component or more precisely its Steiner points). After a full component is chosen, it is added to G_K. The full component is also added to the temporary solution in loss-contracted form. This ensures that components which are in conflict with accepted ones are less appealing subsequent iterations. Finally, a *MST* is built on the union of G_K and all chosen full components. By these means, the algorithm achieves an approximation ratio of 1.55 if $k \to \infty$ and is computable in $\mathcal{O}(|K|^k \cdot |V - K|^{k-2} + k \cdot |K|^{2k+1} \log |K|)$ (Robins and Zelikovsky 2005).

A k-restricted full component F is a full component with $k \geq 3$ terminals. By $C_l[F]$, we denote the loss-contracted full component of F. We define the gain and loss of a full component F formally and then describe the execution of Algorithm 2.

Definition 5 (*Gain and Loss of a Full Component* (Robins and Zelikovsky 2005)) Let T be a tree spanning K and F be an arbitrary full component of G given K.

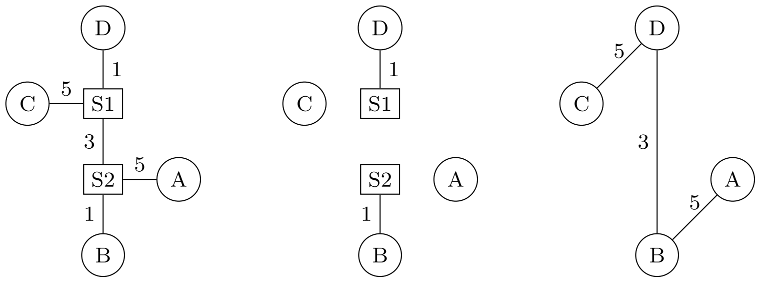
Let $T[F]$ be a minimum cost graph in $F \cup T$ which contains F completely and spans all terminals in K. This means $T[F]$ is the result of replacing a part of the tree T with the full component F.

Then, the Gain of F w.r.t. T is the cost difference between T and $T[F]$:

$$gain_T(F) = cost(T) - cost(T[F])$$

The loss of F is a minimum-cost subforest of F containing a path from each Steiner point in F to one of its terminals: $Loss(F_t) = MST\big(F_t \cup E_0(F_t)\big) \backslash E_0(F_t)$, where $E_0(F)$ denotes a complete graph containing all terminals of F, with all edge costs being 0. It follows that

$$loss(F) = cost\Big(MST\big(F \cup E_0(F)\big) \backslash E_0(F)\Big)$$



(a) Full component **(b)** Loss of full component **(c)** Contracted full component

Fig. 7 Full components

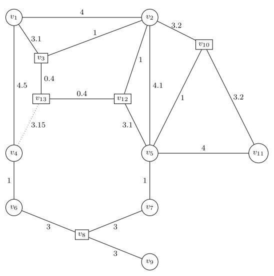
```
    **Data:** 2-connected graph $G = (V, E, b)$, terminal set $K \subseteq V$, an integer $k$ with
          $3 \leq k \leq |K|$
    **Result:** A $k$-restricted Steiner tree $S(G)$ in $G$ spanning $K$.
 1  Compute $G_V$ and $G_K$
 2  $T = MST(G_K)$
 3  **repeat**
 4      Find a $k$-restricted full component $F$ maximizing
 5          $r = gain_T(F)/loss(F)$
 6      $G_K = G_K \cup F$
 7      $T = MST(T \cup C_l[F])$
 8  **until** $\underline{r \leq 0}$;
 9  $S(G, K) = MST(G_K)$
10  Replace artificial edges in $S(G, K)$
11  Cut Steiner point leaves of $S(G, K)$
12  **return** $S(G, K)$
```

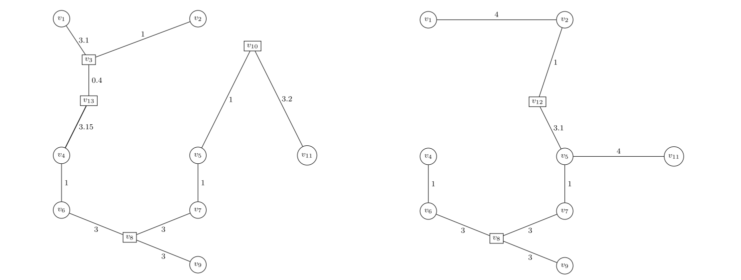
Algorithm 2: Approximation Allocation Algorithm A^{RZ}

The algorithm starts by computing G_V, its subgraph G_K (Line 1) and the *MST* on G_K (Line 2). Afterwards, the gain-over-loss ratios for all k-restricted full components are computed. It is sufficient to consider k-restricted full components consisting of k terminals and between 1 and $k - 2$ Steiner points since every component is uniquely identified by its Steiner points of degree larger than 2 (Fig. 7a). Note that the gain of a full component F is dependent on T, while the loss is not. After choosing the full component with the highest gain-over-loss ratio, the selected component is added to G_K (Line 6). The component is also added to T in loss-contracted form $C_l[F]$ (Line 7).

To contract the loss of a full component F, we merge every connected tree of the forest $Loss(F)$ into a single vertex, the respective terminal of the component. Two terminals are connected in $C_l[F]$ if their respective components in $Loss(F)$ have an

Fig. 8 G





(a) solution for G where the owner of $\{V4, V13\}$ bids 3.15 **(b)** solution for G' where the owner of $\{V4, V13\}$ bids 3.11

Fig. 9 RZ: solutions **a** solution for G where the owner of $\{V4, V13\}$ bids 3.15 **b** solution for G' where the owner of $\{V4, V13\}$ bids 3.11

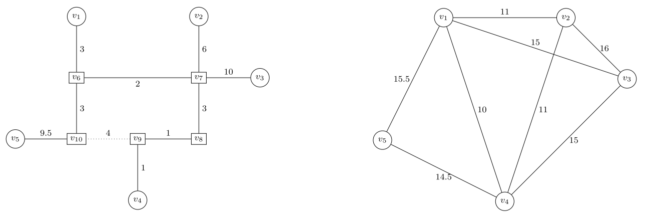
adjacent edge in F (Fig. 7b) and the cost of the edge in $C_l[F]$ is equal to the cost of the respective edge in F (Fig. 7c).

After $C_l[F]$ was added to T, an MST is built on $T \cup C_l[F]$. By improving T, the gain-over-loss ratio for the remaining full components is decreasing. Eventually, all components will have a gain-over-loss ratio of at most zero. At this point, the algorithm computes the $MST(G_K)$ (Line 9), transforms all its artificial edges back into original edges, i.e. replaces artificial edges by the respective shortest path, and cuts leaves which are Steiner points (Line 11).

Proposition 5 *Allocation algorithm A^{RZ} is not monotonic.*

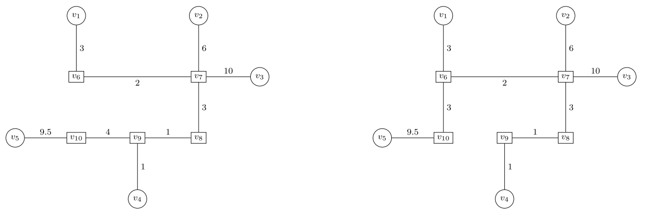
Proof The proof is by counter example. Consider graph G (Fig. 8) with terminal set $K = \{v_1, v_2, v_4, v_5, v_6, v_7, v_9, v_{11}\}$ (round nodes) and the non-terminals $\{v_3, v_8, v_{10}, v_{12}, v_{13}\}$ (rectangular nodes). The owner of $e = \{v_4, v_{13}\}$ bids 3.15.

The computed solution of cost 22.85 can be seen in Fig. 9a. Note that $e = \{v_4, v_{13}\}$ is part of the solution.



(a) G **(b)** relevant edges of G_K

Fig. 10 G and distance network G_K



(a) solution for G where the owner of $\{V9, V10\}$ bids 4 **(b)** solution for G where the owner of $\{v_9, v_{10}\}$ bids 3

Fig. 11 MST-based solutions. **a** Solution for G where the owner of $\{V9, V10\}$ bids 4 **b** solution for G where the owner of $\{v_9, v_{10}\}$ bids 3

However, if the owner of e reduces her bid to 3.11, the solution does not include e any more (Fig. 9b). Thus, monotonicity is violated and we cannot hope to achieve strategyproofness by using a critical payment scheme. □

So, the algorithm by Robins and Zelikovsky (2005) based on loss-contraction cannot easily be extended to a strategyproof mechanism. Let us next analyze approximation algorithms based on the distance network.

Distance-network-based approximations

Similarly to the loss-contracting approximation, the general idea of distance-network-based approximation algorithms is to build a *MST* on a complete subgraph G_K in the first phase. In the second phase, edges (shortest paths) in the *MST* are decomposed into edges in E, and a *MST* is computed on the resulting graph to remove possible cycles. Finally, in the third phase, non-terminal leaves are deleted. This algorithm was proposed by Kou et al. and runs in $\mathcal{O}(|K||V|^2)$. However, due to the cycles that can occur in the first phase, this standard variant is not monotonic. To see this, consider the graph G in Fig. 10 with its relevant edges of the respective distance network G_K. If the bid for $e = \{v_9, v_{10}\}$ is 4, e is part of the solution (Fig. 11a). However, if the bid was only 3, e might be removed from the solution (Fig. 11b). Hence, monotonicity is violated.

Gualà and Proietti (2005) change the algorithm in its second phase when the MST on the subgraph G_K is replaced by the corresponding shortest paths. Instead of adding all shortest paths and afterwards calculating the MST on the resulting graph to remove the cycles, in the extended algorithm the shortest paths are inserted iteratively in a way such that no cycles are introduced. The authors show that such an acyclic expansion is always possible. The resulting algorithm requires a runtime of $\mathcal{O}((|V| + |K|^2)|E| \cdot log\ \alpha(|E|, |V|))$ where $\alpha(.,.)$ is the classic inverse of the Ackermann's function as defined in Tarjan (1982) and yields a $2(1 - 1/|K|)$-approximation.

Mehlhorn (1988) designed an algorithm which differs in phase 1. Here, the algorithm first partitions G into Voronoi regions, which are then utilized to construct a subgraph of G_K, called $\overline{G}$. It then proceeds with phase 2 and phase 3 as described above. This leads to a worst case runtime of $\mathcal{O}(|V| \log |V| + |E|)$ and achieves an approximation ratio of $2(1 - 1/l)$ where l is the minimal number of leaves in any minimum Steiner tree (which is naturally bounded above by the number of terminals). In the following, we discuss Mehlhorn's algorithm and show that the allocation is monotonic. Hence, the algorithm is also suitable to be extended to an approximation mechanism with a slightly better runtime and approximation ratio than other algorithms based on distance-network-based approximation algorithms.

Definition 6 (*Voronoi Regions $\mathcal{V}(s)$*) Given a general graph $G = (V, E, b)$ and the set of terminals $K \subseteq V$, the *Voronoi region $\mathcal{V}(s)$* of a terminal $s \in K$ contains all vertices $v \in V$ for which the shortest path $sp(s, v) \leq sp(t, v)$ for all $t \in K$. We break ties randomly, such that each vertex v uniquely belongs to one such region.

Definition 7 (*Distance network based on $\mathcal{V}$*) Let $\bar{G} = (K, E_{\bar{G}}, b_{\bar{G}})$ be the distance network with edges and weights as follows:

$$(s,t) \in E_{\bar{G}} \Leftrightarrow \exists\, (u,v) \in E \text{ such that } u \in \mathcal{V}(s) \text{ and } v \in \mathcal{V}(t)$$
$$b_{\bar{G}}(s,t) = \min\{sp(s,u) + b(u,v) + sp(v,t) \,:\, u \in \mathcal{V}(s),\ v \in \mathcal{V}(t),\ (u,v) \in E\}$$

```
  Data: 2-connected graph $G = (V, E, b)$, terminal set $K \subseteq V$
  Result: A Steiner tree $S(G, K)$ in $G$ spanning $K$
1 Compute Voronoi regions of $G$ and generate $\bar{G}$;
2 $S(G, K) = MST(\bar{G})$
3 Replace artificial edges in $S(G, K)$
4 Cut non-terminal leaves of $S(G, K)$
  return $S(G, K)$
```

Algorithm 3: Approximation Allocation Algorithm $\mathcal{A}^{MH}$

Similar to the algorithm by Gualà and Proietti (2005), the algorithm by Mehlhorn is also monotonic.

Proposition 6 *The allocation of Mehlhorn's algorithm is monotonic.*

Proof Suppose there is a graph G, an allocation A computed by Mehlhorn's algorithm and an edge $e \in A$. Further assume that the owner of e lowers her bid. Reducing a bid for an edge e can only mean that e is now part of at least as many shortest paths as before. In general, this may change the allocation. However, only shortest paths which contain e are cheaper after the changed bid. Hence, e will remain part of the solution even though it might be chosen in another path. It remains to be shown that a changed allocation cannot lead to cycles in the solution and thus to the possible exclusion of e.

Suppose there is a cycle between two Voronoi regions. This would mean that two paths between the respective regions have been chosen. Since an *MST* is built on the subgraph induced by the Voronoi regions, this can never happen during Mehlhorn's algorithm. A similar argument holds for cycles in more than two Voronoi regions. Finally, there can be no cycles inside a single Voronoi region (by definition). Since no cycle can occur, no edge that has been added to the solution will be removed from the solution at a later point and therefore e is in the final solution. Thus, the allocation computed by Mehlhorn's algorithm is monotonic. □

Primal-dual approximation algorithms

This section describes the general approach for primal-dual approximations and the approximation algorithm for the minimum Steiner tree problem by Goemans and Williamson (1995) which requires a runtime of $O(|V|^2 \log |V|)$ and also has an approximation ratio of 2.

Many problems in graph theory can be reduced to the hitting set problem. For a ground-set E with cost $c_e \geq 0$ for every element $e \in E$ and subsets $T_1, T_2 \dots T_n \subseteq E$, the hitting set problem is to find a subset $A \subseteq E$ of minimal cost such that $A \cap T_i \neq \emptyset$ for all subsets $i = \{1, \dots n\}$. The primal integer program for the hitting set problem can be formulated as follows:

$$\begin{array}{lll} \text{Min} & \sum_{e \in E} c_e x_e & \\ \text{subject to} & \sum_{e \in T_i} x_e \geq 1, & \forall i \\ & x_e \in \{0, 1\}. & \end{array}$$

To obtain the relaxation, simply the constraint $x_e \in \{0, 1\}$ needs to be relaxed to $x_e \geq 0$. The corresponding dual program is stated below:

$$\begin{aligned} \text{Max} \quad & \sum_i y_i \\ \text{subject to} \quad & \sum_{i:e\in T_i} y_i \le c_e, \ \forall e \in E \\ & y_i \ge 0, \ \ \forall i. \end{aligned}$$

To obtain an α-approximation, we compute a solution $\bar{x}$ to the primal integer program and a solution y to the dual of the relaxed primal program such that $\sum_{e\in E} c_e \bar{x}_e \le \alpha \sum_{i=1}^{n} y_i$.

```
    Data: ground-set $E$, subsets $T_1, T_2, \dots T_n \subseteq E$
    Result: allocation $A$
1   $y = 0 \ \ \forall y$
2   $A = \emptyset$
3   while $A$ not feasible do
4       Find violated $T_k$ $(T_k \cap A = \emptyset)$
5       Increase $y_k$ until $\exists e \in T_k$ s.t. $\sum_{i:e\in T_i} y_i = c_e$
6       $A = A \cup \{e\}$
7   end
8   return $A$
```

Algorithm 4: Approximation Algorithm for the hitting set problem

Algorithm 4 describes the necessary steps to compute A. During the initialization, A is empty and all dual variables y are set to 0. In each iteration, a violated set T_k is chosen. Afterwards, the corresponding dual variable y_k is increased (loaded) until one of the constraints holds with equality (it goes "tight," Line 5). The corresponding element e is then added to the solution. If the allocation A is feasible, the algorithm stops and returns A.

Mapping the hitting set problem to the minimum Steiner tree problem is straightforward: The ground-set is given by the edges E of the graph and c_e is the cost of the respective edge $e \in E$. Let S_i be a subset of vertices that contains at least one, but not all terminals, i.e. a cut. When all cuts are crossed, the solution is a feasible allocation for the minimum Steiner tree problem. By definition, the edges adjacent to exactly one vertex $v \in S_i$ are the edges crossing the cut S_i. Let $\delta(S_i)$ denote the set of these edges. Let $T_i = \delta(S_i)$. The adapted algorithm can be seen below (Algorithm 5). It achieves an approximation ratio of 2 (Goemans and Williamson 1995).

```
Data: 2-connected graph $G = (V, E, b)$, terminal set $K \subseteq V$
Result: A Steiner tree $S(G)$ in $G$ spanning $K$
1  $y = 0 \ \forall y$
2  $A_0 = \emptyset$
3  $i = 0$
4  while $\underline{A_i \text{ not feasible}}$ do
5      Choose violated sets $\mathcal{U}$
6      Increase $y_k$ uniformly for all $T_k \in \mathcal{U}$ until $\exists e_i \notin A_i$ s.t. $\sum_{i:e_i \in T_i} y_i = c_{e_i}$
7      $A_i = A_i \cup \{e_i\}$
8      $i = i + 1$
9  end
10 $A' = A_{i-1}$
11 for $\underline{i;\ i \geq 0;\ i = i - 1}$ do
12     if $\underline{A' \setminus \{e_{t_i}\} \text{ still feasible}}$ then
13         $A' = A' \setminus \{e_{t_i}\}$
14     end
15 end
```

Algorithm 5: Approximation Allocation Algorithm $\mathcal{A}^{PD}$

Two modifications can be seen in Algorithm 5 in comparison with the basic primal-dual algorithm (Algorithm 4). Firstly, load is not increased on one, but multiple (minimal) unsatisfied components $T_k \in \mathcal{U}$. $\mathcal{U}$ contains all T_k that are unsatisfied and minimal, i.e. there is no unsatisfied set T_j with $T_j \subset T_k$. Secondly, after computing the allocation A a reverse deletion is conducted. In this phase, edges are assessed in regard to their necessity in reversed order (LIFO). Unnecessary edges either connect a Steiner point as a leaf or close a cycle. In either case, the edge is not contributing to the solution (apart from inflicting costs).

Proposition 7 *The allocation of the primal-dual based* minimum Steiner tree *approximation is monotonic.*

Proof Suppose there is a graph G, an allocation A computed by the primal-dual approximation algorithm for Steiner trees and an edge $e \in A$ whose cost has been truthfully stated by its owner. Further assume that the owner of e lowers her bid to c'_e. Due to the lower cost, e can go tight only sooner. Since e was part of the first allocation, we know that conflicting edges have been removed before e was candidate for removal. Since e is now cheaper and was thus added to the solution earlier or at the same point, it still is considered for removal later than the conflicting edges. Hence, when e is assessed for necessity, the conflicting edges have already been removed. The allocation computed by the primal-dual approximation algorithm for Steiner trees is thus monotonic. □

References

Archer A, Tardos E (2001) Truthful mechanisms for one-parameter agents. In: Proceedings of the 42nd IEEE symposium on foundations of computer science. IEEE Computer Society, Washington, DC, USA, FOCS '01, pp 482–491

Bilò D, Forlizzi L, Gualà L, Proietti G (2007) An algorithm composition scheme preserving monotonicity. In: Proceedings of the twenty-sixth annual ACM symposium on principles of distributed computing. ACM, New York, NY, USA, PODC '07, pp 360–361

Buchsbaum AL, Kaplan H, Rogers A, Westbrook JR (1998) Linear-time pointer-machine algorithms for least common ancestors, MST verification, and dominators. In: Proceedings of the thirtieth annual ACM symposium on theory of computing, ACM, pp 279–288

Byrka J, Grandoni F, Rothvoß T, Sanità L (2010) An improved LP-based approximation for steiner tree. In: Proceedings of the forty-second ACM symposium on theory of computing. ACM, New York, NY, USA, STOC '10, pp 583–592. https://doi.org/10.1145/1806689.1806769

Clarke E (1971) Multipart pricing of public goods. Public Choice XI:17–33

Contreras I, Fernández E (2012) General network design: a unified view of combined location and network design problems. Eur J Oper Res 219(3):680–697

Dütting P, Gkatzelis V, Roughgarden T (2017) The performance of deferred-acceptance auctions. Math Oper Res 42(4):897–914

Erickson RE, Monma CL, Veinott Jr AF (1987) Send-and-split method for minimum-concave-cost network flows. Math Oper Res 12(4):634–664

Gkatzelis V, Markakis E, Roughgarden T (2017) Deferred-acceptance auctions for multiple levels of service. In: Proceedings of the 2017 ACM conference on economics and computation. ACM, pp 21–38

Goemans MX, Williamson DP (1995) The primal-dual method for approximation algorithms and its application to network design problems. In: Hochbaum D (eds) Approximation algorithms for NP-hard problems. PWS Publishing, Boston, pp 144–191

Green J, Laffont JJ (1977) Characterization of satisfactory mechanisms for the revelation of preferences for public goods. Econometrica J Econ Soc 45(2):427–438

Groves T (1973) Incentives in teams. Econometrica 41:617–631

Gualà L, Proietti G (2005) A truthful (2 - 2/k)-approximation mechanism for the steiner tree problem with k terminals*. In: Proceedings of the 11th annual international conference on computing and combinatorics. Springer, Berlin, COCOON'05, pp 390–400. https://doi.org/10.1007/11533719_40

Iwata Y, Shigemura T (2018) Steiner tree solver [software]. https://github.com/wata-orz/steiner_tree. Accessed 8 May 2018

Karp R (1972) Reducibility among combinatorial problems. In: Miller R, Thatcher J (eds) Complexity of computer computations. Plenum Press, Berlin, pp 85–103

Koch T, Martin A, Voß S (2000) SteinLib: An updated library on steiner tree problems in graphs. Technical report ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin. http://elib.zib.de/steinlib. Accessed 12 Mar 2018

Kou L, Markowsky G, Berman L (1981) A fast algorithm for steiner trees. Acta Informatica 15(2):141–145. https://doi.org/10.1007/BF00288961

Kourtellis N, Morales GDF, Bonchi F (2015) Scalable online betweenness centrality in evolving graphs. IEEE Trans Knowl Data Eng 27(9):2494–2506

Lavi R, Swamy C (2011) Truthful and near-optimal mechanism design via linear programming. J ACM 58(6):25:1–25:24. https://doi.org/10.1145/2049697.2049699

Lehmann D, O'Callaghan LI, Shoham Y (2002) Truth revelation in approximately efficient combinatorial auctions. J ACM 49(5):577–602. https://doi.org/10.1145/585265.585266

Leyton-Brown K, Milgrom P, Segal I (2017) Economics and computer science of a radio spectrum reallocation. Proc Natl Acad Sci 114(28):7202–7209

Mehlhorn K (1988) A faster approximation algorithm for the steiner problem in graphs. Inf Process Lett 27(3):125–128

Milgrom P, Segal I (2019) Clock auctions and radio spectrum reallocation. J Polit Econ. https://doi.org/10.1086/704074

Mu'alem A, Nisan N (2008) Truthful approximation mechanisms for restricted combinatorial auctions. Games Econ Behav 64(2):612–631. https://doi.org/10.1016/j.geb.2007.12.009

Myerson RB (1981) Optimal auction design. Math Oper Res 6(1):58–73

Newman N, Leyton-Brown K, Milgrom P, Segal I (2017) Assessing economic outcomes in simulated reverse clock auctions for radio spectrum. CoRR abs/1706.04324. arxiv:1706.04324

Nisan N, Ronen A (1999) Algorithmic mechanism design (extended abstract). In: Proceedings of the thirty-first annual ACM symposium on theory of computing. ACM, New York, NY, USA, STOC '99, pp 129–140. https://doi.org/10.1145/301250.301287

Nisan N, Roughgarden T, Tardos E, Vazirani V (2007) Algorithmic game theory. Cambridge University Press, Cambridge

Öncan T, Cordeau JF, Laporte G (2008) A tabu search heuristic for the generalized minimum spanning tree problem. Eur J Oper Res 191(2):306–319

Pettie S, Ramachandran V (2002) Computing shortest paths with comparisons and additions. In: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, pp 267–276

Robins G, Zelikovsky A (2005) Tighter bounds for graph steiner tree approximation. SIAM J Discrete Math 19(1):122–134. https://doi.org/10.1137/S0895480101393155

Takahashi H, Matsuyama A (1980) An approximate solution for the steiner problem in graphs. MathJaponica 24:573–577

Tarjan RE (1974) A note on finding the bridges of a graph. Inf Process Lett 2:160–161

Tarjan RE (1982) Sensitivity analysis of minimum spanning trees and shortest path trees. Inf Process Lett 14(1):30–33

Vazirani VV (2013) Approximation algorithms. Springer, Berlin

Vickrey W (1961) Counterspeculation, auctions, and competitive sealed tenders. J Finance 16(1):8–37

Xu J, Chiu SY, Glover F (1995) Tabu search heuristics for designing a steiner tree based digital line network. University of Colorado, Boulder

Affiliations

Martin Bichler[1] · Zhen Hao[1] · Richard Littmann[1,2] · Stefan Waldherr[1]

Zhen Hao
zhen.hao@in.tum.de

Richard Littmann
richard.littmann@in.tum.de

Stefan Waldherr
stefan.waldherr@in.tum.de

[1] Technical University of Munich, Munich, Germany

[2] Advanced Optimization in a Networked Economy (AdONE), GRK 2201, Technical University of Munich, Leopoldstr. 145, 80804 Munich, Germany