Algorithmica (2007) 49: 171–191
DOI 10.1007/s00453-007-9016-7

Exact and Approximate Truthful Mechanisms for the Shortest Paths Tree Problem

Luciano Gualà · Guido Proietti

Published online: 14 September 2007

Abstract Let a communication network be modeled by an undirected graph $G = (V, E)$ of n nodes and m edges, and assume that edges are controlled by selfish agents. In this paper we analyze the problem of designing a truthful mechanism for computing one of the most popular structures in communication networks, i.e., the *single-source shortest paths tree*.

More precisely, we will study several realistic scenarios, in which each agent can own either a single or multiple edges of G. In particular, for the single-edge case, we will show that: (i) in the classic utilitarian case, the problem can be solved efficiently in $O(mn \log \alpha(m, n))$ time, where $\alpha(m, n)$ is the inverse of the Ackermann's function; (ii) in a meaningful non-utilitarian case, namely that in which agents' valuation functions only depend on the edge lengths, the problem can be solved in $O(m + n \log n)$ time. Conversely, for the multiple-edges case, we will show in the utilitarian case an $O(mP + nP \log n)$ time truthful mechanism, where $P = O(n)$ denotes the number of agents participating in the solution, while in the same non-utilitarian case we will prove a general lower bound to the approximation ratio that can be achieved by any truthful mechanism, by showing that no c-approximate mechanism can exist, for any fixed $c < \frac{5+\sqrt{13}}{3+\sqrt{13}}$.

Work partially supported by the Research Project GRID.IT, funded by the Italian Ministry of Education, University and Research. Part of the results herein contained was presented at the 11th International Euro-Par Conference (Euro-Par'05), Lisbon, Portugal, 2005.

L. Gualà (✉) · G. Proietti
Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, 67010 L'Aquila, Italy
e-mail: guala@di.univaq.it

L. Gualà
Dipartimento di Matematica, Università di Tor Vergata, 00133 Rome, Italy

G. Proietti
Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", CNR, Viale Manzoni 30, 00185 Rome, Italy
e-mail: proietti@di.univaq.it

Keywords Single-source shortest paths tree · Selfish agents · Algorithmic mechanism design · Truthful mechanisms

1 Introduction

Mechanisms are a classical concept in the theory of non-cooperative games [17]. In these games there are several independent agents which privately hold part of the data input, and that have to interact with the system in order to build a solution optimizing a given system-wide global objective function. However, each agent has her own intrinsic benefit or loss in participating in any specific solution (called her *valuation* function), and may speculate about her data input in the hope of getting a higher profit. This leads to economically suboptimal resource allocation and is therefore undesirable. The main objective of *mechanism design* theory is then to study how to incentivize the agents—by means of suitable *payments*—in order to cooperate with the system. Correspondingly, a *mechanism* is the coupling of an algorithm computing a feasible solution with a payment scheme specifying the payments provided to the agents. Informally, a mechanism is *truthful* if its payments guarantee that agents are not encouraged to lie. Then, the problem of combining the game theoretic concept of designing a truthful mechanism, with the computational complexity requirement of designing an efficient algorithm, is the focus of the *algorithmic mechanism design* (AMD) for selfish agents.

In their seminal paper concerned with AMD [16], Nisan and Ronen addressed the classic *shortest path* problem, in which it is given an undirected graph $G = (V, E)$ of n nodes and m edges where each edge is owned by a selfish agent, which privately holds the edge length, and the system-wide goal is that of computing a shortest path in G between two given endnodes. As soon as, quite naturally, one assumes that the agent's valuation function is proportional to the owned edge length (obviously once that the edge is part of the solution), then this problem enjoys the property of being *utilitarian*, in that the quality of any feasible output can be measured by simply summing up all the agents' valuations. For utilitarian problems, there exists a well-known class of truthful mechanisms, i.e., the *Vickrey-Clarke-Groves (VCG) mechanisms* [5, 8, 21], and therefore the shortest path problem can be solved efficiently (in terms of runtime for computing the output specification and the payments to the agents) in $O(m + n \log n)$ time [10, 11]. In contrast with VCG-mechanisms, which handle arbitrary valuation functions but only utilitarian problems, Archer and Tardos [1] defined another class of truthful mechanisms, named *one-parameter mechanisms*, allowing to solve general (i.e., *non-utilitarian*) problems, but with the restriction that the data input held by each agent must be expressed by a single parameter. Network design non-utilitarian problems occur frequently, since cases in which the solution must optimize some objective which cannot be expressed as the sum of the agents' valuations abound. By exploiting the results in [1, 16], in a sequence of papers, efficient truthful mechanisms have been designed for solving several network design problems such as: the shortest path problem for the case in which agents own nodes [14], the *minimum spanning tree* problem [15], the NP-hard *minimum Steiner tree* problem [9], and the non-utilitarian *minimum radius spanning tree* problem [19].

In this paper, we focus on the problem of computing one of the most popular network topologies, that is the *single-source shortest paths tree* (SPT), in the setting

in which each agent can own one or more edges of the underlying graph. What is interesting here is that an SPT naturally admits both utilitarian and non-utilitarian formulations. Indeed, as we will discuss later in the paper, it can well happen that an agent gives an evaluation of her contribution which is simply proportional to her private input, and this unavoidably makes the problem non-utilitarian. Therefore, we analyze both scenarios, and we provide the following main results:

- In the utilitarian case, we provide a VCG-mechanism which can be implemented: (i) for the special *single-edge case* (i.e., that in which each agent owns a single edge), in $O(mn \log \alpha(m, n))$ time on a RAM, and in $O(mn\,\alpha(m, n))$ time on a pointer machine, where $\alpha(m, n)$ is the inverse of the Ackermann's function defined in [20]; (ii) otherwise, for the general *multiple-edges case* (i.e., that in which each agent owns multiple edges), in $O(mP + nP \log n)$ time, where $P = O(n)$ denotes the number of agents participating in the solution.
- In the prominent non-utilitarian case in which the agent's valuation function is restricted to depend only on the length of each owned edge, we provide both negative and positive results. On the negative side, for the multiple-edges case, we prove a general lower bound of $\frac{5+\sqrt{13}}{3+\sqrt{13}}$ to the approximation ratio that can be achieved by any truthful mechanism, and moreover we show that no c-approximate *additive mechanism* (see Definition 4) can exist, for any fixed $c > 1$. Concerning positive results, we give: (i) for the special single-edge case, an exact $O(m + n \log n)$ time one-parameter mechanism; (ii) otherwise, for the general multiple-edges case, an n-approximate VCG-mechanism which can be implemented in almost optimal $O(m\,\alpha(m, n))$ time, and an exact $O(mP + nP \log n)$ time mechanism *with verification* [16].

Notice that for the single-edge case, the non-utilitarian case can be solved more efficiently than the utilitarian one, and essentially this is due to the fact that our specific non-utilitarian case allows for a faster computation of the payments. Conversely, for the multiple-edges case, the utilitarian case can be solved exactly through a VCG-mechanism, while a solution for the non-utilitarian case can only be approximate, since no multiple-parameter truthful mechanism general techniques are known to date.

The paper is organized as follows: In Sect. 2 we recall some basic definitions from both graph theory and algorithmic mechanism design; in Sect. 3 we deal with the utilitarian version of our problem, while in Sect. 4 we analyze the different solutions for the non-utilitarian case. Finally, Sect. 5 lists some open problems.

2 Basic Definitions

2.1 Graph Notation

Let $G = (V, E)$ be an undirected graph, with $|V| = n$ nodes and $|E| = m$ edges, and with a positive real weight associated with each edge $e \in E$. For a given subgraph H of G, we will denote by $E(H)$ the set of edges of H, unless otherwise specified. Given a source node s and a destination node z in G, a path in G between s and z

is a *shortest path*, say $P_G(s, z)$, if the sum of its edge weights (called *distance* in G between s and z, and denoted by $d_G(s, z)$) is minimum. Given a source node $s \in V$, we denote by $S_G(s)$ a *single-source shortest paths tree* (SPT) of G rooted at s. Notice that shortest paths are not unique, in general, and whilst for VCG-mechanisms this lack of uniqueness is irrelevant, for one-parameter mechanisms the truthfulness can be guaranteed only if a tie-breaking rule is adopted, as observed in [1]. For the sake of uniformity, we therefore assume that nodes of G are numbered arbitrarily, and if we can lead to a node by using different shortest paths, then we choose that in which the predecessor of such a node has minimum index. Given $u, v \in V$, we denote by $\mathrm{LCA}(u, v)$ the *least common ancestor* of u and v in $S_G(s)$, i.e., the ancestor of both u and v in $S_G(s)$ which is farthest from s.

Let $e = (u, v) \in E(S_G(s))$ be a tree edge, with u closer to s than v. Let $M(e)$ denote the set of nodes in $S_G(s)$ reachable from s without passing through edge e, and let $N(e) = V \backslash M(e)$ be the remaining nodes. Sets $M(e)$ and $N(e)$ define a cut in G, and $C(e) = \{f = (x, y) \in E \backslash \{e\} \mid (x \in M(e)) \wedge (y \in N(e))\}$ is the set of edges *crossing* the cut. Moreover, we denote by $||e||$ the cardinality of $N(e)$.

2.2 Mechanism Design

Let a communication network be modeled by a graph $G = (V, E)$, and assume that edges of G are owned by selfish agents. We denote by $\mathcal{A} = \{a_1, \ldots, a_N\}$ the set of agents, and we assume that no agent is *necessary*, i.e., we assume that the removal of all the edges belonging to any agent leaves non-empty the set of feasible solutions for the problem we are solving. In graph-theoretic terms, this means that if we denote by E_i the set of edges owned by any $a_i \in \mathcal{A}$, then the graph $G - a_i = (V, E \backslash E_i)$ is connected. Quite obviously, we assume that for any $a_i, a_j \in \mathcal{A}, i \neq j$, we have that $E_i \cap E_j = \emptyset$.

Each agent a_i holds a private information t_e for each owned edge $e \in E_i$. We call this value the *true type* of the edge e. This value depends on various factors (e.g., bandwidth, reliability, etc.). Only agent a_i knows t_e, while everything else is public knowledge. Each agent has to declare to the mechanism a public *reported type* r_e (also called her *bid*) for each edge $e \in E_i$. In the rest of the paper, for the sake of avoiding technicalities, we will assume that $r_e > 0, \forall e \in E$. With t_i and r_i we will denote the type vector and the reported vector of the agent a_i, respectively. We will denote by t the vector of true types, and by r the vector of bids.

For a given optimization problem defined on G, there exists some set of feasible solutions F that the mechanism is allowed to choose. For each feasible solution $x \in F$, some measure function $\mu(t, x)$ is defined, which depends on the true types. The mechanism tries to optimize $\mu(t, x)$, but of course it does not know t directly.

For every agent a_i, a function $v_i(t_i, x)$ expresses a_i's *valuation* with respect to an output $x \in F$: this is a quantification of the contribution of a_i to x. While t_i is known only by the agent a_i, the valuation function is public. In order to offset the costs deriving from these services, the mechanism provides some reward to agents participating in the computed solution, i.e., the mechanism makes a *payment* $p_i(r)$ to the agent a_i for the service provided in a solution which is computed as a function of the reported vector r.

A *mechanism* is then a pair $\mathcal{M} = \langle g(r), p(r) \rangle$, where $g(r)$ is an algorithm that, given agents' reported types, computes a feasible solution in F, and $p(r)$ is a scheme which describes the payments provided to the agents. For each agent a_i and for each solution $g(r)$ computed by the mechanism, the *utility* function of a_i is defined as $u_i(t_i, r) = p_i(r) - v_i(t_i, g(r))$. We assume that each agent is selfish, i.e., she always attempts to maximize her utility. Let r_{-i} denote the vector of all bids besides r_i; the pair (r_{-i}, r_i) will denote the vector r. We say that *truth-telling* is a *dominant strategy* for agent a_i if bidding t_i always maximizes her utility, regardless of what the other agents bid, i.e., $u_i(t_i, (r_{-i}, t_i)) \geq u_i(t_i, (r_{-i}, r_i))$, for all r_{-i} and r_i. A mechanism is said to be *truthful* if, for every agent, truth-telling is a dominant strategy. Moreover, let $\varepsilon(\sigma)$ denote a positive real function of the input size σ. Then, an $\varepsilon(\sigma)$-*approximate mechanism* is a mechanism which returns a solution $g(r)$ which comes within a factor $\varepsilon(\sigma)$ from the optimum, i.e., $\mu(t, g(r)) \leq \varepsilon(\sigma) \cdot \mu(t, x^*)$, where x^* is an optimal solution with respect to the vector t. We say that a mechanism is *poly-time computable* if $g(\cdot)$ and $p(\cdot)$ are computable in polynomial time, and that it satisfies the *voluntary participation* condition if agents never incur in a net loss (i.e., such that the agents' utilities are always non-negative).

One of the most important results in mechanism design theory is the class of the well-known *Vickrey-Clarke-Groves* (VCG) mechanisms. A VCG-mechanism applies to mechanism design problems called *utilitarians* and enjoys the fundamental property of being truthful. A mechanism design problem is called *utilitarian* if its measure function satisfies $\mu(t, x) = \sum_{a_i \in \mathcal{A}} v_i(t_i, x)$.

Definition 1 (VCG-mechanisms) A mechanism is of the *VCG-family* if:

1. $g(r) \in \arg\min_{x \in F} \left\{ \sum_{a_i \in \mathcal{A}} v_i(r_i, x) \right\}$;
2. For any $a_i \in \mathcal{A}$, let $h_i(r_{-i})$ be an arbitrary function independent of r_i, and let $\mathcal{A}_{-i} = \mathcal{A} \setminus \{a_i\}$; then, the payment for a_i is

$$p_i(r) = h_i(r_{-i}) - \sum_{a_j \in \mathcal{A}_{-i}} v_j(r_j, g(r)).$$

Basically, VCG-mechanisms handle arbitrary valuation functions, but only utilitarian problems. In [1], Archer and Tardos have shown how to design truthful mechanisms for non-utilitarian problems under the assumption that the problem is *one-parameter*. A problem is called one-parameter if (i) the type of each agent a_i can be expressed as a single parameter $t_i \in \mathbb{R}$, and (ii) each agent's valuation has the form $v_i(t_i, x) = t_i\, w_i(r)$, where $w_i(r)$ is called *work curve* for agent a_i, i.e., some amount of work that depends on the algorithmic output specification x, which in turn is a function of the reported types vector r. In [1], it is shown that a mechanism for a one-parameter problem is truthful if and only if it belongs to the following class of mechanisms:

Definition 2 (One-parameter mechanisms) A mechanism is *one-parameter* if $g(r)$ is non-increasing (i.e., $w_i(r_{-i}, r_i)$ is a non-increasing function of r_i, for all $a_i \in \mathcal{A}$, and

all r_{-i}), and the payment provided to any agent a_i has the form:

$$p_i(r_{-i}, r_i) = h_i(r_{-i}) + r_i w_i(r_{-i}, r_i) - \int_0^{r_i} w_i(r_{-i}, z)\,\mathrm{d}z, \tag{1}$$

where $h_i(r_{-i})$ is an arbitrary function independent of r_i.

Moreover, in [1] it is shown that if $\int_0^{+\infty} w_i(r_{-i}, z)\,\mathrm{d}z < +\infty$ for all a_i and all r_{-i}, then we can use the following payment scheme to obtain a truthful mechanism satisfying also the voluntary participation condition

$$p_i(r_{-i}, r_i) = r_i\, w_i(r_{-i}, r_i) + \int_{r_i}^{+\infty} w_i(r_{-i}, z)\,\mathrm{d}z. \tag{2}$$

3 The Utilitarian Case

Let a communication network be modeled by an undirected graph $G = (V, E)$ in which subsets of edges of E are owned by selfish agents. In the following, we will denote by G and G^t the input graph as weighted with respect to the reported values and to the true types, respectively.

Suppose that a_i holds, as the private type t_e for each owned edge $e \in E_i$, the length of the communication link, and thus the time needed to cross it, and assume that the system-wide goal is to minimize the completion time for delivering a message from a distinguished node $s \in V$ to every node $v \in V \setminus \{s\}$. This means that the system looks for an SPT rooted at s of G^t.

3.1 A VCG-Mechanism

By using the notation introduced in the previous section, the problem can be formalized as follows. The set of feasible solutions F is the set of all the spanning trees (considered in the following as rooted at s) of G^t, and the measure of a solution $T \in F$ is

$$\mu(t, T) = \sum_{v \in V} d_T(s, v). \tag{3}$$

To complete the description of the problem, we have to define the agents' valuation. It is clear that, if an agent a_i participates in the output with an edge $e \in E_i$, she will incur in a transmission cost (i.e., the cost for forwarding a message through that edge). In our scenario it is reasonable to assume that the transmission cost is proportional to the length of the edge, i.e., proportional to the value t_e. Notice that the TCP/IP protocol used in Internet for broadcasting a message is the so-called *unicast*. In this protocol, if a source wants to send a message to a set of recipients, it must send a copy of the message for each destination. Therefore, if any solution $T \in F$ is used for broadcasting a message from s to all the other nodes, then the cost for the agent a_i can be expressed as follows:

$$v_i(t_i, T) = \sum_{e \in E_i} v_e(t_e, T),$$

where, with a small abuse of notation, with $v_e(t_e, T)$ we denote the valuation function of a_i with respect to edge $e \in E_i$, which is equal to

$$v_e(t_e, T) = \begin{cases} t_e\|e\| & \text{if } e \in E(T); \\ 0 & \text{otherwise.} \end{cases}$$

Indeed, if an agent a_i participates in the output T, she will incur a transmission cost of t_e for each edge $e \in E(T) \cap E_i$ and for each message which passes through e, and the number of these messages is exactly $\|e\|$.

From the above assumptions, it immediately follows that the problem is utilitarian. Indeed, the measure function (3) can be rewritten as

$$\mu(t, T) = \sum_{v \in V} d_T(s, v) = \sum_{e \in E(T)} t_e\|e\| = \sum_{e \in E} v_e(t_e, T)$$
$$= \sum_{i=1}^{N} \sum_{e \in E_i} v_e(t_e, T) = \sum_{i=1}^{N} v_i(t_i, T).$$

This means that we can use a VCG-mechanism to solve the utilitarian SPT problem. Therefore, let $\mathcal{M}_1$ be a mechanism in this class defined as follows:

1. The algorithmic output specification selects an SPT $S_G(s) = (V, E')$ of G;
2. Let $G - a_i = (V, E \backslash E_i)$. Then, the payment function for a_i is defined as[1]

$$p_i(r) = \sum_{v \in V} d_{G-a_i}(s, v) - \left(\mu(r, S_G(s)) - \sum_{e \in E' \cap E_i} r_e\|e\| \right). \tag{4}$$

Notice that the above payments obey Definition 1, since the term $\sum_{v \in V} d_{G-a_i}(s, v)$ corresponds to $h_i(r_{-i})$, while the term in parenthesis corresponds to

$$\mu(r, S_G(s)) - \sum_{e \in E' \cap E_i} r_e\|e\| = \sum_{e \in E'} r_e\|e\| - \sum_{e \in E' \cap E_i} r_e\|e\|$$
$$= \sum_{e \in E' \backslash E_i} v_e(r_e, S_G(s)) = \sum_{a_j \in \mathcal{A}_{-i}} v_j(r_j, g(r)).$$

Hence, the mechanism is a (truthful) VCG-mechanism. Furthermore, the above payment scheme (more precisely, the fact that $h_i(r_{-i})$ is set to be equal to the measure of an optimal solution in the graph $G - a_i$) induces a so-called *pivotal mechanism*, which can be shown to satisfy the voluntary participation condition [5].

3.2 Mechanism Time Complexity

The algorithmic question is now the following: how fast can the above mechanism be computed?

[1] Recall that from the assumption that no agent is necessary, graph $G - a_i$ is connected, and therefore $d_{G-a_i}(s, v)$ is bounded for any $v \in V$; also recall that $G - a_i$ is weighted with respect to the reported types, and thus distances are computed according to these weights.

3.2.1 The Multiple-Edges Case

First of all, the SPT $S_G(s)$ can be found in $O(m + n \log n)$ time [6]. On the other hand, to compute $p_i(r)$ for each $a_i \in \mathcal{A}$ participating in the solution (an agent which participates with no edge in the solution clearly receives a payment equal to 0), we start by observing that the term in parenthesis in (4) can be found in $O(n)$ time for all the agents, by means of a modified post-order visit of $S_G(s)$ in which at each node v we maintain both the total length of all the paths emanating from v towards its descendants in $S_G(s)$ (in order to compute $\mu(r, S_G(s))$), and the size of the subtree of $S_G(s)$ rooted at v (in order to compute $\|e = (u, v)\|$). Thus, it remains to find all the distances $d_{G-a_i}(s, v)$, for every $v \in V$. A brute-force solution consists of computing a new SPT of the graph $G - a_i$ from scratch. In this way, if we denote by $P \leq \min\{N, n-1\}$ the number of agents participating in the solution, we have that computing all the payments takes $O(mP + nP \log n)$ time. In the worst case, $P = n - 1$, and thus the mechanism requires $O(mn + n^2 \log n)$ time. Improving this bound would require to circumvent the bottleneck of recomputing from scratch all the SPTs, but unfortunately this sounds as hard as the dynamic SPT problem, and thus we left the problem of beating the trivial upper bound open for further research. However, for the special notable case in which each agent controls only a single edge, we show in the following how to improve (on a RAM) the above bound to $O(mn \log \alpha(m, n))$ time. Notice that the best improvement is for sparse graphs (i.e., for $m = \Theta(n)$), where it amounts to an almost logarithmic factor.

3.2.2 The Single-Edge Case

Let then us assume that for each $a_i \in \mathcal{A}$, we have $|E_i| = 1$. This means, there are m agents, each owning a distinct edge of G. In the sequel, for the purpose of simplifying the notation, the appropriate functions will be indexed by using e. With these assumptions, if we denote by $G - e$ the graph G after the removal of edge e, the payment scheme of the mechanism $\mathcal{M}_1$ can be rewritten as follows:

$$p_e(r) = \begin{cases} \sum_{v \in V} d_{G-e}(s, v) - (\mu(r, S_G(s)) - r_e \|e\|) & \text{if } e \in E'; \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Once again, to compute $p_e(r)$ for each $e \in E'$, the bottleneck is to find all the distances $d_{G-e}(s, v)$, for every $v \in V$. As before, a brute-force solution, consisting in computing for each edge $e \in E'$ a new SPT of the graph $G - e$ from scratch, would take $O(mn + n^2 \log n)$ time. Thus, to improve this performance, we need to adopt a different strategy, as explained in the following.

At a very high-level, our method works as follows. We start by computing, for all the pairs $v, v' \in V$, the distance $d_G(v, v')$. Then, we solve $n - 1$ subproblems. Each subproblem is identified by a distinct destination node $z \in V \setminus \{s\}$, and involves computing the distance $d_{G-e}(s, z)$ for each edge e of the path $P_G(s, z)$ in $S_G(s)$ between s and z. The solutions of these subproblems will be then properly composed to find, for each $e \in E'$, all the distances $d_{G-e}(s, v)$, for every $v \in V$.

Let e be an edge on $P_G(s,z)$, and let $P_{G-e}(s,z)$ be a replacement shortest path for e, i.e., a path from s to z in $G-e$ of (minimum) length $d_{G-e}(s,z)$. The problem of finding all the replacement shortest paths, one for each edge of $P_G(s,z)$, has been efficiently solved in $O(m+n\log n)$ time on a pointer machine [11], and in $O(m\,\alpha(m,n))$ time on a word RAM [12], respectively. Both algorithms are based on a pre-computation of the SPTs $S_G(s)$ and $S_G(z)$. We now show how to improve (on a RAM) the above results to $O(m\log\alpha(m,n))$ time (once that $S_G(s)$ and $S_G(z)$ are given):

Proposition 1 *Let $P_G(s,z)$ be a shortest path joining s and z, and assume that for any $v\in V$, distances $d_G(s,v)$ and $d_G(z,v)$ are given. Then, the set of distances $D(s,z)=\{d_{G-e}(s,z)\mid e\in E(P_G(s,z))\}$ can be computed on a RAM in $O(m\log\alpha(m,n))$ time.*

Proof Since a replacement shortest path $P_{G-e}(s,z)$ joining s and z must contain an edge in $C(e)$, it follows that it corresponds to a path of length

$$d_{G-e}(s,z)=\min_{f=(x,y)\in C(e)}\big\{d_{G-e}(s,x)+r_f+d_{G-e}(y,z)\big\}. \tag{6}$$

Observe that $d_{G-e}(s,x)=d_G(s,x)$, since x is reachable in $S_G(s)$ from s without passing through e. Concerning $d_{G-e}(y,z)$, the following holds (see also [13]):

Lemma 1 *Let $e=(u,u')$ be an edge on $P_G(s,z)$, with u closer to s than u', and let $f=(x,y)\in C(e)$. Then, y is reachable in $S_G(z)$ from z without passing through e, and thus $d_{G-e}(y,z)=d_G(y,z)$.*

Proof Suppose, for the sake of contradiction, that the claim is false. Then, y is a descendant of both u and u' in $S_G(z)$. This means that $P_G(z,y)$ makes use of e, and so we have (since subpaths of shortest paths are shortest paths) that $P_G(u',y)$ is a subpath of $P_G(z,y)$. Therefore, from the assumption that $r_e>0,\forall e\in E$, we have

$$d_G(u',y)=r_e+d_G(u,y)>d_G(u,y).$$

On the other hand, since y is reachable in $S_G(s)$ from s by passing through e, we have that $P_G(s,y)$ makes use of e, and so we have that $P_G(u,y)$ is a subpath of $P_G(s,y)$. Hence

$$d_G(u,y)=r_e+d_G(u',y)>d_G(u',y),$$

that is, we have a contradiction. □

From the above lemma, it follows that (6) is equivalent to

$$\begin{aligned} d_{G-e}(s,z)&=\min_{f\in C(e)}\big\{d_{S_G(s)}(s,x)+r_f+d_{S_G(z)}(y,z)\big\}\\ &=\min_{f\in C(e)}\big\{\ell(f):=d_G(s,x)+r_f+d_G(y,z)\big\}. \end{aligned} \tag{7}$$

Since distances in G from s and z are given in input, it follows that $\ell(f)$ (called the *label* of edge f) is available in $O(1)$ time for any given $f \in C(e)$. Thus, finding $d_{G-e}(s,z)$ reduces to select an edge associated with e, say f_e, having minimum label over all the $O(m)$ edges of $C(e)$. However, since there are $O(n)$ edges in $P_G(s,z)$, this means that an exhaustive search would find $D(s,z)$ in $O(mn)$ time. Thus, we need a different approach. To this respect, we make use of a *Split-Findmin* structure [7]. This is a data structure operating on a collection of disjoint sequences of elements. Initially, there is only one sequence containing all the elements, and each element j has a key $\kappa(j) := +\infty$. Then, the structure supports the following operations:

`split`(j): Split the sequence containing j into two sequences of elements: one up to and including j, the other sequence taking the rest;
`findmin`(j): Return the element (and the associated key) in j's sequence with minimum key;
`decrease-key`(j, K): Set $\kappa(j) := \min\{\kappa(j), K\}$.

For our purposes, we initialize a Split-Findmin structure, say $\mathcal{Q}$, in which the initial sequence consists of the n vertices of $S_G(s)$, as sorted in any arbitrary post-order, each with key $+\infty$. After, we associate with each non-tree edge f the corresponding label $\ell(f)$, and we associate with each node u of $P_G(s,z)$, the set of non-tree edges $F(u) = \{f = (x,y) \in E \setminus E' \mid \mathrm{LCA}(x,y) = u\}$. We then consider all the edges in $P_G(s,z)$, one after the other, in the order of their occurrence starting from s. Let $\langle (u_0 = s, u_1), (u_1, u_2), \ldots, (u_{q-1}, u_q = z)\rangle$ denote this sequence of edges of $P_G(s,z)$. Throughout the execution of this procedure, two invariants in $\mathcal{Q}$ are maintained:

($\mathcal{I}_1$): Every sequence in $\mathcal{Q}$ corresponds to some rooted subtree of $S_G(s)$ (the initial sequence is associated with $S_G(s)$);
($\mathcal{I}_2$): When edge (u_{i-1}, u_i) is considered, $i = 1, \ldots, q$, it holds that for any $y \in V$ in the subtree of $S_G(s)$ rooted at u_i, key $\kappa(y)$ corresponds to the label of a minimum-label edge (not in $S_G(s)$, by definition) connecting y to a vertex outside the current y's sequence (i.e., outside the subtree of $S_G(s)$ currently containing y).

Let us now see how $\mathcal{Q}$ is used. We start by considering the edge $e_1 = (s, u_1)$. Because of the post-order arrangement of the nodes, s is the rightmost element in the initial n-element sequence. Then, we perform one split centered at the element preceding s in the sequence (this will sever s), and one additional split (in any arbitrary order) for each of the children of s in $S_G(s)$, to reestablish invariant ($\mathcal{I}_1$). After, we focus on the sequence associated with the child of s in $P_G(s,z)$, namely u_1, and we restore invariant ($\mathcal{I}_2$) by performing a number of decrease-key operations. More precisely, we consider all the edges $f = (x,y) \in F(s)$ such that y is a descendant of u_1 in $S_G(s)$, and we issue the operation `decrease-key`$(y, \ell(f))$. Afterwards, we perform a `findmin`(u_1) operation, which will return a key $\kappa(u_1) = \ell(f_{e_1})$, where f_{e_1} is an edge belonging to $P_{G-e_1}(s,z)$, and $\ell(f_{e_1})$ is exactly the distance $d_{G-e_1}(s,z)$. Figure 1 depicts this first step of the procedure.

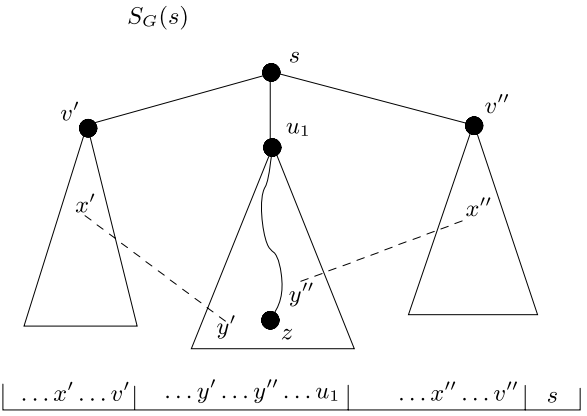

Fig. 1 The initial sequence corresponding to $S_G(s)$ is split after considering edge $e_1 = (s, u_1)$. *Dashed edges* are those for which a decrease-key operation is performed. Eventually, a `findmin`(u_1) operation will return an edge minimizing (7) with respect to e_1

At the generic ith step, edge (u_{i-1}, u_i) is considered, and operations in $\mathcal{Q}$ are performed as described above, by letting u_{i-1} and u_i now taking the place of s and u_1, respectively. This process goes ahead until the last edge of $P_G(s, z)$ is considered.

Let us now analyze the time complexity of this procedure. Since $\ell(f)$ is available in $O(1)$ time for a fixed non-tree edge f, labeling all the non-tree edges takes $O(m)$ time. Concerning the Split-Findmin operations, in total there are $O(m)$ operations: $O(n)$ splits (one for each subtree whose root is adjacent to some node of $P_G(s, z)$), $O(n)$ findmins (one for each node of $P_G(s, z)$), and $O(m)$ decrease-keys (at most one for each non-tree edge). This takes on a RAM $O(m \log \alpha(m, n))$ time [18]. Other costs, such as the post-order traversal and finding least common ancestors, are linear [2]. From this, the claim follows. □

We are now ready to prove the main result of this section:

Theorem 1 *For the special case in which each agent owns a single edge of G, mechanism $\mathcal{M}_1$ can be computed on a RAM in $O(mn \log \alpha(m, n))$ time.*

Proof As observed before, the output specification can be computed in $O(m + n \log n)$ time [6]. As far as the payment scheme is concerned, we proceed as follows. First, we find the all-pairs distances in G in $O(mn \log \alpha(m, n))$ time [18], and then we solve each of the $n - 1$ above described subproblems in $O(m \log \alpha(m, n))$ time. Then, for each edge $e = (u, v) \in E'$, we extract in $O(n)$ time from the solutions of the subproblems all the distances $d_{G-e}(s, x)$, for every x in the subtree of $S_G(s)$ rooted at v (all the other nodes clearly maintain their distance from s in $G - e$). After this step, $p_e(r)$ can be found in $O(n)$ time, since as already shown before, $\mu(r, S_G(s)) - r_e \|e\|$ can be obtained for all the agents in $O(n)$ time. Since we have to compute exactly $n - 1$ payment functions, one for each tree edge, the claim follows. □

Notice that on a pure pointer machine model, under the same assumptions of Theorem 1, the mechanism $\mathcal{M}_1$ can be computed in $O(mn\,\alpha(m, n))$ time, since in this case the Split-Findmin data structure requires $O(m\,\alpha(m, n))$ time for solving any given subproblem [18].

4 A Meaningful Non-Utilitarian Case

The utilitarian scenario assumes that each agent, in doing her valuation, starts from the assumption that each atomic operation will involve a traffic load on the owned edge which is proportional to the edge length times the size of the corresponding appended subtree of $S_G(s)$. However, in another reasonable scenario, an agent might evaluate her participation to an output $T \in F$ simply as follows:

$$v_i(t_i, T) = \sum_{e \in E_i} v_e(t_e, T),$$

where

$$v_e(t_e, T) = \begin{cases} t_e & \text{if } e \in E(T); \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

This scenario is realistic whenever the agent starts from the assumption that each atomic operation will involve a traffic load on the owned edge which is proportional only to the edge length (this can happen, for instance, when the transmission protocol replicates at each node a given message once for each descending node, like in the *Internet Protocol multicast* [4], so that each tree edge will simply afford the cost of forwarding a single message).

This setting makes the problem non-utilitarian, since the measure function associated with the SPT problem does not equal the sum of the agents' valuations. In the following, we show how to approach the problem from different perspectives. In Sect. 4.1 we provide an exact one-parameter truthful mechanism for the particular case in which each agent owns a single edge, while in Sect. 4.2 we address the general case and (i) we prove lower bounds to the approximation ratio that can be achieved by any truthful mechanism; (ii) we provide an approximate VCG-mechanism, and (iii) we design an exact truthful mechanism with verification, a weaker model of mechanism introduced in [12]. In either case we make use of the pointer machine computational model, since we cannot take advantage of the addressing capabilities of a RAM, as we did for the utilitarian case. For the sake of brevity, we will refer to our non-utilitarian SPT problem as to *the* non-utilitarian case, though several non-utilitarian settings can in principle be postulated.

4.1 A One-Parameter Mechanism for the Single-Edge Case

Consider the case in which each agent controls a single edge of G. As in Sect. 3.2.2, we simplify the notation by indexing the appropriate functions by e, and we assume that the agents apply a direct valuation, and thus the valuation of an agent a_e has the form as in (8). Let now $g(r)$ denote the output specification of an algorithm computing an SPT of G. It is easy to see that in this case the problem can be solved through a one-parameter mechanism [1], since for each agent a_e, we can rewrite the valuation (8) as $v_e(t_e, g(r)) = t_e\, w_e(r)$, where

$$w_e(r) = \begin{cases} 1 & \text{if } e \in E(g(r)); \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Indeed, for each agent a_e, if we denote by θ_e the maximum reported type for e such that e belongs to the computed solution (the so-called *threshold* value), then the function $w_e(r_{-e}, r_e)$ is equal to 1 for $0 \leq r_e \leq \theta_e$, and is equal to 0 for any $r_e > \theta_e$, which implies that $g(r)$ is non-increasing. Moreover, as far as the payment scheme is concerned, observe that for the single-edge case, the assumption we did that no agent is necessary, means that G is 2-edge-connected (i.e., there are no bridges). Consequently, the maximum reported type for being part of a solution is bounded. From this, the integral $\int_0^{+\infty} w_e(r_{-e}, z)\,\mathrm{d}z$ is bounded for all the edges, and then we can apply the payment (2) for an agent a_e participating in the solution (for all other agents, the payment is obviously equal to 0), which can now be rewritten as

$$p_e(r_{-e}, r_e) = r_e w_e(r_{-e}, r_e) + \int_{r_e}^{+\infty} w_e(r_{-e}, z)\,\mathrm{d}z = r_e + \theta_e - r_e = \theta_e.$$

Then, by applying the general definition of a one-parameter mechanism [1], and by using the payment scheme (2), we obtain the following mechanism $\mathcal{M}_2$ for the non-utilitarian SPT problem:

1. The algorithmic output specification selects an SPT $S_G(s) = (V, E')$ of G;
2. The payment function for a_e is defined as the threshold θ_e, if $e \in E'$, and 0 otherwise.

We can now prove the following result:

Theorem 2 *The mechanism $\mathcal{M}_2$ is a truthful mechanism satisfying the voluntary participation condition for the single-edge non-utilitarian SPT problem, and it can be computed on a pointer machine in $O(m + n \log n)$ time.*

Proof The truthfulness follows from the fact that $\mathcal{M}_2$ is a one-parameter mechanism as we showed above. Moreover, since we use the payment scheme (2), the voluntary participation condition is guaranteed as well.

From the time complexity point of view, once again the output specification can be computed in $O(m + n \log n)$ time. Concerning the payments, we have to compute all the thresholds θ_e, for each $e \in E'$. Let now $e = (u, v) \in E'$, with u closer to s that v. Then, e remains in $S_G(s)$ as long as $d_G(s, u) + r_e \leq d_{G-e}(s, v)$, from which it follows that $\theta_e = d_{G-e}(s, v) - d_G(s, u)$. As shown in [13], computing $d_{G-e}(s, v)$ is equivalent to selecting a non-tree edge such that

$$d_{G-e}(s, v) = \min_{f=(x,y)\in C(e)} \big\{ d_{G-e}(s, x) + r_f + d_{G-e}(y, v) \big\}. \tag{10}$$

The selection of all the non-tree edges (one for each tree edge) satisfying (10) costs $O(m\,\alpha(m, n))$ time [13]. This means that we can compute all the payments in $O(m\,\alpha(m, n)) = O(m + n \log n)$ time, from which the claim follows. □

4.2 On the Complexity of the Multiple-Edges Case

In this section, we analyze the case in which each agent can own multiple edges, by reporting both positive and negative results as far as the solvability of the problem is concerned.

4.2.1 A General Lower Bound

Before proving a general lower bound to the approximation ratio that any truthful mechanism can achieve, we recall two basic properties of any truthful mechanism. For any agent $a_i \in \mathcal{A}$, let us denote by $g_i(t)$ the set $E_i \cap E(T = g(t))$. Moreover, given a set $X \subseteq E_i$ of edges, we will use $t(X)$ to denote $\sum_{e \in X} t_e$. Notice that $t(g_i(t))$ is equal to a_i's valuation $v_i(t_i, g(t))$. The following two claims are essentially quoted from [16].

Claim 1 (Independence) *Let $\mathcal{M} = \langle g(\cdot), p(\cdot)\rangle$ be a truthful mechanism for the non-utilitarian SPT problem, let t and t' be type vectors, and let a_i be an agent. If $t_{-i} = t'_{-i}$ and $g_i(t) = g_i(t')$, then $p_i(t) = p_i(t')$.*

From the above claim, given a truthful mechanism, we can represent the payment that the mechanism returns to an agent by using the following well defined function.

Definition 3 Let $\mathcal{M} = \langle g(\cdot), p(\cdot)\rangle$ be a truthful mechanism for the non-utilitarian SPT problem, let t be a type vector, and let a_i be an agent. For a set $X \subseteq E_i$ of edges, we define *the payment returned to a_i for X* as:

$$p_i(X, t_{-i}) = \begin{cases} p_i(t_{-i}, t'_i) & \text{if } \exists t'_i \text{ s.t. } X = g_i(t_{-i}, t'_i); \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Claim 2 (Maximization) *Let $\mathcal{M} = \langle g(\cdot), p(\cdot)\rangle$ be a truthful mechanism for the non-utilitarian SPT problem, let t be a type vector, and let $X_t = g_i(t)$. Then, for each agent a_i, we have*

$$p_i(X_t, t_{-i}) - t(X_t) \geq p(X, t_{-i}) - t(X), \quad \forall X \subseteq E_i. \tag{12}$$

Intuitively, Claim 2 states that, for each agent a_i, a truthful mechanism must choose a subset $X_t \subseteq E_i$ of edges which maximizes a_i's utility.[2] We can now prove our general lower bound. We start by showing the following

Proposition 2 *Let k be any real value larger than or equal to 2. Then, there does not exist any c-approximate truthful mechanism for the non-utilitarian SPT problem when agents own multiple edges, for any fixed $c < L(k) = \min\{\frac{3k}{2k+1}, 1 + \frac{1}{k}\}$, even in the special case in which each agent owns a subset of edges incident to a node.*

Proof Let $0 < \epsilon < 1/k$, and let M be a suitably large value. Consider the situation in Fig. 2(a), where each edge e is labeled with its true type value t_e. Suppose that an agent, say a_1, owns the set of edges $\{e_1, e_2\}$, which are both incident to s, while the remaining edges are arbitrarily owned by other agents. Now, for the sake of contradiction, suppose $\mathcal{M}$ is a truthful c-approximate mechanism, with $c < L(k)$. Then, since

[2] Note that in the non-utilitarian scenario, the valuation of an agent a_i depends only on the subset $X \subseteq E_i$ of edges that the mechanism selects in the solution. Thus, it is easy to see that the formula $p(X, t_{-i}) - t(X)$ in (12) represents the utility of a_i with respect to any solution T such that $E(T) \cap E_i = X$.

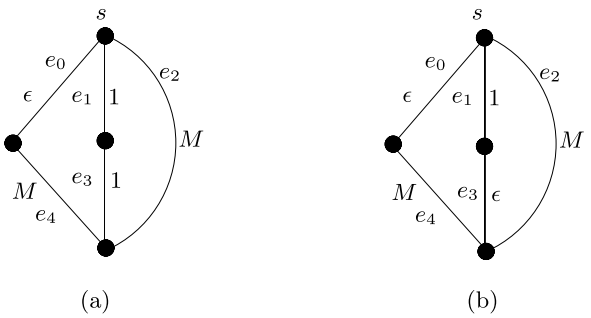

Fig. 2 The two instances used to prove the lower bounds

$L(k) \leq 3/2$ for every $k \geq 2$, the solution selected by $\mathcal{M}$ must be $T = (V, \{e_0, e_1, e_3\})$ (otherwise $\mathcal{M}$ cannot be c-approximate for M large enough).

Let now $\Delta p = p_1(\{e_1, e_2\}, t_{-1}) - p_1(\{e_1\}, t_{-1})$. The following lemma holds:

Lemma 2 $\Delta p \geq 1 + 1/k$.

Proof For the sake of contradiction, suppose $\Delta p < 1 + 1/k$, and consider the following new type profile $t' = (t_{-1}, t'_{e_1} = t_{e_1}, t'_{e_2} = 1 + 1/k)$. Since t_{-1} is not changed, from Claim 1 we have that all the payments returned to a_1 remain the same. Then, from Claim 2, $\mathcal{M}$ must select the output that maximizes a_1's utility, and it is easy to see that $\mathcal{M}$ will still select T. Indeed, if e_2 enters in the solution then a_1's utility becomes strictly lower, since a_1 incurs in an additional cost of $1 + 1/k$, but she receives only an additional payment of $\Delta p < 1 + 1/k$.

Let $\mu = \mu(t', T)$ be the measure of the solution computed by the mechanism, and let μ_{OPT} be the measure of an optimal solution. It is easy to see that $\mu = 3 + \epsilon$ and $\mu_{\mathrm{OPT}} = 2 + 1/k + \epsilon$. It follows that the approximation ratio achieved by the mechanism is

$$\rho = \frac{\mu}{\mu_{\mathrm{OPT}}} = \frac{3k + k\epsilon}{2k + 1 + k\epsilon}$$

which goes to $\frac{3k}{2k+1}$ for ϵ that goes to 0. Since $c < L(k) \leq \frac{3k}{2k+1}$, we obtain a contradiction, since we can always choose ϵ small enough so that $\rho > c$ (and thus, $\mathcal{M}$ cannot be c-approximate), and the lemma follows. □

Now consider the following new type profile $t'' = (t_{-1}, t''_{e_1} = \epsilon, t''_{e_2} = 1 + \frac{1}{k} - \epsilon)$. Once again, t_{-1} is not changed and thus all the payments returned to a_1 remain the same. From Lemma 2 we have that $\Delta p \geq 1 + 1/k$. Thus, from Claim 2, $\mathcal{M}$ must select a solution which maximizes a_1's utility, and then such a solution must be $T' = (V, \{e_0, e_1, e_2\})$. Clearly, for ϵ small enough, the optimal solution is $T^* = (V, \{e_0, e_1, e_3\})$. Thus, the approximation ratio achieved by the mechanism is

$$\rho = \frac{\mu(t'', T')}{\mu(t'', T^*)} = \frac{\epsilon + 1 + 1/k}{3\epsilon + 1}$$

which goes to $1 + 1/k$ for ϵ that goes to 0. Since $c < L(k) \leq 1 + 1/k$, we obtain a contradiction, since we can always choose ϵ small enough so that $\rho > c$ (and thus, $\mathcal{M}$ cannot be c-approximate). From this, the claim follows. □

From the above proposition, we immediately have the following:

Theorem 3 *There does not exist any c-approximate truthful mechanism for the non-utilitarian SPT problem when agents own multiple edges, for any fixed* $c < \frac{5+\sqrt{13}}{3+\sqrt{13}}$, *even in the case in which each agent owns a subset of edges incident to a node.*

Proof The claim follows from Proposition 2 by choosing k in order to maximize $L(k) = \min\left\{\frac{3k}{2k+1}, 1 + \frac{1}{k}\right\}$. Let $f(k) = \frac{3k}{2k+1}$ and $g(k) = 1 + 1/k$. A trivial computation shows that $L(k)$ is maximum when $f(k) = g(k)$, i.e., when $k = \bar{k} = \frac{3+\sqrt{13}}{2}$. Thus we obtain a lower bound of $L(\bar{k}) = f(\bar{k}) = g(\bar{k}) = \frac{5+\sqrt{13}}{3+\sqrt{13}}$. □

4.2.2 A Tighter Lower Bound for Additive Mechanisms

We now use similar arguments to prove a tighter lower bound for a particular class of mechanisms, named *additive* mechanisms [16]:

Definition 4 A mechanism is called *additive* if for each agent a_i, each type profile t and each set $X \subseteq E_i$ of edges, $p_i(X, t_{-i}) = \sum_{e \in X} p_i(\{e\}, t_{-i})$.

Theorem 4 *Any additive truthful mechanism for the non-utilitarian SPT problem when agents own multiple edges, can be forced to compute a solution which is arbitrarily far from the optimal one.*

Proof Let $k \geq 1$, let $0 < \epsilon < 1/k$, and let M be a suitably large value. Consider the situation in Fig. 2(b), where each edge e is labeled with its true type value t_e. Suppose that an agent, say a_1, owns the set of edges $\{e_1, e_2\}$, while the remaining edges are owned by other agents in an arbitrary way. The proof is by contradiction. Let $\mathcal{M}$ be an additive truthful c-approximate mechanism, for some constant $c > 1$. Then, for M large enough, the solution selected by $\mathcal{M}$ must be $T = (V, \{e_0, e_1, e_3\})$.

The following lemma holds:

Lemma 3 *For any positive real value k, we have* $p_1(\{e_2\}, t_{-1}) \geq 1/k$.

Proof Suppose $p_1(\{e_2\}, t_{-1}) < 1/k$, and consider the new type profile $t' = (t_{-1}, t'_{e_1} = 1 - \epsilon, t'_{e_2} = 1/k)$. Since t_{-1} has not changed, from Claim 1 once again all the payments returned to a_1 remain the same. It is easy to see that the solution which maximizes a_1's utility (and that the mechanism must select from Claim 2) is still T, while, for ϵ small enough, the optimal one is $T^* = (V, \{e_0, e_2, e_3\})$. Thus, the ratio between the corresponding measures is

$$\frac{\mu(t', T)}{\mu(t', T^*)} = \frac{2k}{2\epsilon k + 2}$$

which goes to k for ϵ that goes to 0. This contradicts the fact that $\mathcal{M}$ is a c-approximate mechanism, since we can choose k arbitrarily large. □

Now consider a new type profile $t'' = (t_{-1}, t''_{e_1} = \epsilon, t''_{e_2} = \frac{1}{k} - \epsilon)$. It is easy to see that a_1 obtains a strictly positive additional utility from e_1 (since $p_1(\{e_1\}, t_{-1})$ is unchanged and $t''_{e_1} < t_{e_1}$), and from e_2 (from Lemma 3). Then, the solution computed by the mechanism must contain both e_1 and e_2, and thus such solution will be $T' = (V, \{e_0, e_1, e_2\})$. On the other hand, for ϵ small enough, the optimal solution is $T^* = (V, \{e_0, e_1, e_3\})$. It follows that

$$\frac{\mu(t'', T')}{\mu(t'', T^*)} = \frac{1 + k\epsilon}{4k\epsilon}$$

which is unbounded for ϵ that goes to 0. Clearly, this is once again a contradiction, and the theorem follows. □

4.2.3 An Approximate VCG-Mechanism

To design an approximate mechanism for the multiple-edges non-utilitarian SPT problem, a brute-force solution consists of choosing a VCG-mechanism. Since the algorithmic output specification has to minimize the sum of the agents' valuations, this will clearly return a *minimum spanning tree* (MST) of G^t. Given a tree T, let $w(T) = \sum_{e \in E(T)} r_e$ denote the total weight of T. More formally, let $\mathcal{M}_3$ be the mechanism defined as follows:

1. The algorithmic output specification selects an MST $T_G = (V, E')$ of G, and root it at s;
2. Let T_{G-a_i} be an MST of $G - a_i$. Then, the payment function for an agent a_i is defined as

$$p_i(r) = w(T_{G-a_i}) - \left(w(T_G) - \sum_{e \in E' \cap E_i} r_e \right).$$

Theorem 5 *The mechanism $\mathcal{M}_3$ is a truthful n-approximate mechanism satisfying the voluntary participation condition for the multiple-edges non-utilitarian SPT problem, and it can be computed on a pointer machine in $O(m P \alpha(m, n))$ time, where P is the number of agents participating in the solution. In the special case in which each agent controls a subset of edges incident to a node, the mechanism runtime can be lowered to $O(m \alpha(m, n))$.*

Proof It is easy to see that the mechanism belongs to the VCG-family. Indeed, $g(\cdot)$ minimizes the sum of the agents' valuations, and the payments obey Definition 1, since the term $w(T_{G-a_i})$ corresponds to $h_i(r_{-i})$, while the term in parenthesis corresponds to

$$w(T_G) - \sum_{e \in E' \cap E_i} r_e = \sum_{e \in E'} r_e - \sum_{e \in E' \cap E_i} r_e = \sum_{e \in E' \setminus E_i} v_e(r_e, T_G) = \sum_{a_j \in \mathcal{A}_{-i}} v_j(r_j, g(r)).$$

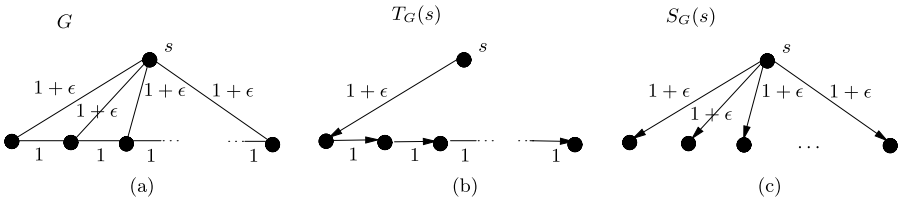

Fig. 3 In (**a**), the input graph G, where $\epsilon > 0$ is arbitrary small; in (**b**) an MST of G rooted at s, which is an $n/2$-approximation of the SPT of $S_G(s)$ depicted in (**c**), for ϵ going to 0

Hence, the mechanism is a pivotal VCG-mechanism. Consequently, T_G is an MST of G^t. Concerning the approximation ratio, let $S_{G^t}(s)$ be an optimal solution for the SPT problem, and let $T_G(s)$ denote the tree T_G once rooted at s. We now show that the solution returned by the VCG-mechanism is a factor-n approximation. Indeed, we have that

$$\mu(t, T_G(s)) = \sum_{e \in E'} t_e \|e\| \leq n \sum_{e \in E'} t_e = n w(T_G) \leq n w(S_{G^t}(s)) \leq n \mu(t, S_{G^t}(s)).$$

Concerning the time complexity, observe that the mechanism essentially requires the computation on a pointer machine of an MST of G, and of an MST of $G - a_i$, for each agent participating in the solution. An MST can be computed in $O(m\,\alpha(m, n))$ time [3]. Thus, since the number of agents participating in the solution is P, the time complexity is $O(m P\,\alpha(m, n))$. Since $P \leq \min\{N, n-1\}$, in the worst case $P = n-1$, and the mechanism runtime is $O(mn\,\alpha(m, n))$. However, whenever each agent owns a subset of edges incident to a node, we can improve the time complexity by using the algorithm in [15] to compute all the trees T_{G-a_i} in $O(m\,\alpha(m, n))$ time, and the claim follows. □

Notice that the above approximation ratio is asymptotically tight, since it is easy to exhibit an example in which an MST is a $\Theta(n)$-approximation of an SPT (see Fig. 3). This means that we cannot hope to get a better approximate result by means of VCG-mechanisms.

4.2.4 An Exact Mechanism with Verification

To overcome the negative results of Sect. 4.2.1 and Sect. 4.2.2, and the limitations of VCG-mechanisms, we follow the approach of the so-called mechanisms with *verification* [16]. In this framework the mechanism is allowed to pay the agents *after* the messages have been broadcasted, knowing the actual *time* (and therefore the corresponding edge's length, once we assume that traveling speed is constant) it was needed for a message to cross each edge in the solution. In the following, for the sake of intuitiveness, we therefore assume that the true type of an agent is, for each owned edge, the corresponding crossing time.

More formally, a mechanism with verification $\mathcal{M}$ works as follows. Given the vector of the agents' reported values r, $\mathcal{M}$ computes a solution (i.e., a tree rooted at s) $g(r)$. Then, agents (in the solution) are observed to broadcast the messages.

We denote as $\tilde{t} = (\tilde{t}_{e_1}, \tilde{t}_{e_2}, \ldots, \tilde{t}_{e_m})$ the actual times spent by the agents to forward the messages through their respective edges. An agent can choose the value r_e in an unrestricted manner for an owned edge e, while, if $e \in E(g(r))$, it must be $\tilde{t}_e \geq t_e$, otherwise $\tilde{t}_e = 0$. Moreover, we assume that the valuation of an agent a_i with respect to a solution T is now depending on $\tilde{t}_i$ (and not on t_i), namely that $v_i(\tilde{t}_i, T) = \sum_{e \in E(T) \cap E_i} \tilde{t}_e$, and that the mechanism hands a payment $p_i(\tilde{t}_i, r)$ to each agent a_i which also depends on $\tilde{t}_i$. Finally, the utility of each agent a_i is defined as the difference between the payment provided by the mechanism and her valuation with respect to the computed solution, i.e., $u_i(\tilde{t}_i, r) = p_i(\tilde{t}_i, r) - v_i(\tilde{t}_i, g(r))$.

With these assumptions, we say that a mechanism with verification is *truthful* if for each agent a_i it is a dominant strategy:

($\mathcal{C}_1$): to report her true type (i.e., $r_i = t_i$);
($\mathcal{C}_2$): for every $e \in E(g(r)) \cap E_i$, to forward a message in minimal time (i.e., $\tilde{t}_e = t_e$).

Now, given a tree T, we remind that $\mu((r_{-i}, \tilde{t}_i), T)$ denotes the measure of T as weighted with respect to $(r_{-i}, \tilde{t}_i)$, i.e., $\mu((r_{-i}, \tilde{t}_i), T) = \sum_{v \in V} d_T(s, v)$, where here distances are defined with respect to the vector $(r_{-i}, \tilde{t}_i)$. Notice that $\mu((r_{-i}, \tilde{t}_i), T)$ depends on $\tilde{t}_i$, but not on the actual execution time of the other agents. We are now ready to define a truthful mechanism with verification [16] for the non-utilitarian SPT problem, say $\mathcal{M}_4$:

1. The algorithmic output specification selects an SPT $S_G(s) = (V, E')$ of G;
2. Let $S_{-i} = S_{G-a_i}(s)$. The payment function for a_i is defined as

$$p_i(\tilde{t}_i, r) = \mu(r, S_{-i}) - \left(\mu\big((r_{-i}, \tilde{t}_i), S_G(s)\big) - \sum_{e \in E' \cap E_i} \tilde{t}_e \right).$$

The following can be proved:

Theorem 6 *The mechanism $\mathcal{M}_4$ is a truthful mechanism with verification satisfying the voluntary participation condition for the multiple-edges non-utilitarian SPT problem, and it can be computed on a pointer machine in $O(mP + nP \log n)$ time, where P is the number of agents participating in the solution.*

Proof Observe that, by definition, for each agent a_i we have that her utility is $\mu(r, S_{-i}) - \mu((r_{-i}, \tilde{t}_i), S_G(s))$ (from which it also follows that the voluntary participation condition is guaranteed). From this, condition ($\mathcal{C}_2$) immediately follows, since the function $\mu(x, S_G(s))$ is decreasing for any x_e in x, and thus a_i maximizes her utility by forwarding the message in minimal time, i.e., by choosing $\tilde{t}_e = t_e$ for each $e \in E' \cap E_i$.

Concerning condition ($\mathcal{C}_1$), let $r' = (r_{-i}, t_i)$ and let $S'_G(s) = g(r')$. Since the term $\mu(r, S_{-i})$ does not depend on r_i, we have to show that, for any (r_{-i}, r_i)

$$-\mu\big((r_{-i}, \tilde{t}_i), S'_G(s)\big) \geq -\mu\big((r_{-i}, \tilde{t}_i), S_G(s)\big). \tag{13}$$

Since condition ($\mathcal{C}_2$) is verified, the inequality (13) can be rewritten as

$$\mu\big((r_{-i}, t_i), S'_G(s)\big) \leq \mu\big((r_{-i}, t_i), S_G(s)\big),$$

which holds from the optimality of $S'_G(s)$ with respect to $r' = (r_{-i}, t_i)$.

Concerning the time complexity, once again both the output specification and S_{-i} can be computed in $O(m + n \log n)$ time, while the remaining term in the payment is computable in linear time. Therefore, since the number of agents for which we have to compute the payments is P, the time complexity is $O(mP + nP \log n)$. Since $P \leq \min\{N, n-1\}$, in the worst case $P = n-1$, and the mechanism runtime becomes $O(mn + n^2 \log n)$. □

5 Future Work

There are several directions to be further investigated. On one hand, a natural open problem is to improve the time complexity of our mechanisms. Indeed, for the analyzed non-utilitarian single-edge case, we have provided a truthful mechanism whose time complexity is not worse than the corresponding centralized problem. Can we do the same in the utilitarian single-edge case? Moreover, can we beat the trivial $O(mn + n^2 \log n)$ time upper bound for the utilitarian multiple-edges case?

On the other hand, an interesting open problem is to better understand the analyzed non-utilitarian multiple-edges case. For this case, we proved a constant lower bound to the approximation ratio that can be achieved by any truthful mechanism. Can we prove a tighter lower bound? Does this problem admit a better than $O(n)$-approximate truthful mechanism? At a first stage, these questions could be addressed in the simplified case in which each agent owns a subset of edges incident to a node.

Finally, another interesting issue could be that of considering more complicated network topologies, in which the distance from the root is only one of the possible parameters to be taken into account, along with other fundamental objectives of the network itself, like its cost. This naturally would imply the extension of the algorithmic mechanism design techniques to multiple criteria problems.

Acknowledgements The authors would like to thank the anonymous referees for the useful suggestions that helped us to improve the quality of the paper, and Davide Bilò for inspiring discussions on the topic.

References

1. Archer, A., Tardos, É.: Truthful mechanisms for one-parameter agents. In: Proc. 42nd IEEE Symp. on Foundations of Computer Science (FOCS'01), pp. 482–491 (2001)
2. Buchsbaum, A.L., Kaplan, H., Rogers, A., Westbrook, J.: Linear-time pointer-machine algorithms for least common ancestors, MST verification, and dominators. In: Proc. 30th ACM Symp. on Theory of Computing (STOC'98), pp. 279–288 (1998)
3. Chazelle, B.: A minimum spanning tree algorithm with inverse-Ackermann time complexity. J. ACM **47**(6), 1028–1047 (2000)
4. Cisco Systems Inc.©: Internetworking Technologies Handbook. Cisco Press (2004)
5. Clarke, E.: Multipart pricing of public goods. Public Choice **8**, 17–33 (1971)
6. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM **34**(3), 596–615 (1987)
7. Gabow, H.N.: A scaling algorithm for weighted matching on general graphs. In: Proc. 26th IEEE Symp. on Foundations of Computer Science (FOCS'85), pp. 90–100 (1985)
8. Groves, T.: Incentives in teams. Econometrica **41**(4), 617–631 (1973)
9. Gualà, L., Proietti, G.: A truthful (2-2/k)-approximation mechanism for the Steiner tree problem with k terminals. In: Proc. 11th Int. Computing and Combinatorics Conference (COCOON'05). Lecture Notes in Computer Science, vol. 3595, pp. 390–400. Springer, New York (2005)

10. Hershberger, J., Suri, S.: Vickrey prices and shortest paths: what is an edge worth? In: Proc. 42nd IEEE Symp. on Foundations of Computer Science (FOCS'01), pp. 252–259 (2001)
11. Malik, K., Mittal, A.K., Gupta, S.K.: The k most vital arcs in the shortest path problem. Oper. Res. Lett. **8**, 223–227 (1989)
12. Nardelli, E., Proietti, G., Widmayer, P.: A faster computation of the most vital edge of a shortest path. Inf. Process. Lett. **79**(2), 81–85 (2001)
13. Nardelli, E., Proietti, G., Widmayer, P.: Swapping a failing edge of a single source shortest paths tree is good and fast. Algorithmica **36**(4), 361–374 (2003)
14. Nardelli, E., Proietti, G., Widmayer, P.: Finding the most vital node of a shortest path. Theor. Comput. Sci. **296**(1), 167–177 (2003)
15. Nardelli, E., Proietti, G., Widmayer, P.: Nearly linear time minimum spanning tree maintenance for transient node failures. Algorithmica **40**(2), 119–132 (2004)
16. Nisan, N., Ronen, A.: Algorithmic mechanism design. Games Econ. Behav. **35**, 166–196 (2001)
17. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
18. Pettie, S., Ramachandran, V.: Computing shortest paths with comparisons and additions. In: Proc. 13th ACM Symp. on Discrete Algorithms (SODA'02), pp. 267–276 (2002)
19. Proietti, G., Widmayer, P.: A truthful mechanism for the non-utilitarian minimum radius spanning tree problem. In: Proc. 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA'05), pp. 195–202 (2005)
20. Tarjan, R.E.: Efficiency of a good but not linear set union algorithm. J. ACM **22**(2), 215–225 (1975)
21. Vickrey, W.: Counterspeculation, auctions and competitive sealed tenders. J. Finance **16**, 8–37 (1961)