Publication II

Lauri Ahlroth, Pekka Orponen, André Schumacher. Approximate Data Aggregation and Prize-Collecting Steiner Trees. Submitted to *Theoretical Computer Science*, 28 pages, under review, March 2012.

Approximate Data Aggregation and Prize-Collecting Steiner Trees

Lauri Ahlroth*, Pekka Orponen, André Schumacher

Department of Information and Computer Science and Helsinki Institute for Information Technology HIIT Aalto University, P.O.Box 15400, FI-00076 Aalto, Finland

Abstract

In the problem of *data aggregation in communication networks*, messages arriving at the nodes of a network are to be forwarded to its root, with the option of being merged with other messages along their transmission path. The goal is to minimise, for a given message sequence, the total sum of the transmission link costs and the delay penalties the messages accrue while waiting for merges. For the online version of the problem, we obtain $\mathcal{O}(\log C_{\mathrm{MST}})$-competitive algorithms covering networks of bounded treewidth and so-called outerterminal planar graphs, where C_{MST} is the cost of a minimum spanning tree in the network. These are the first such results for non-treelike networks. Our general algorithm scheme is based on a relation between the data aggregation problem and the Prize-Collecting Steiner Tree (PCST) problem, and it extends to any other family of networks where the PCST problem can be solved efficiently or admits a fully polynomial-time approximation scheme. An auxiliary result of some independent interest is our exact polynomial-time PCST algorithm for outerterminal graphs.

In an offline setting and on treelike networks, we further improve the bound to $\mathcal{O}(\log C_{\mathrm{maxp}})$, where C_{maxp} is the maximum distance of any node from the root.

Keywords: online algorithm, data aggregation, bounded treewidth, prize-collecting Steiner tree

*Corresponding author. Tel. +358 9 470 25141.

Email addresses: lauri.ahlroth@aalto.fi (Lauri Ahlroth), pekka.orponen@aalto.fi (Pekka Orponen), andre.schumacher@aalto.fi (André Schumacher)

1. Introduction

In the problem of data aggregation in communication networks, messages arriving in the nodes of a network are to be forwarded to its root, and messages accumulated at a given intermediate node may be aggregated and forwarded together paying only a single link cost. However, messages accrue a delay penalty for waiting at a node, and the goal is to minimise the total sum of the link costs and delay penalties for a given message sequence. In this paper we present what we believe are the first online heuristics with provable approximation guarantees for this problem in non-treelike networks, and improve known offline approximation bounds on trees.

The first authors to consider the data aggregation problem in the online optimisation framework were Dooly, Goldman and Scott [12, 13], who addressed specifically the issue of aggregating TCP packet acknowledgements (ACK's) across a single link. They proved, among other things, that balancing the cost of an ACK against the total delay cost of accumulated messages (a version of the "rent-to-buy" heuristic) achieves a competitive ratio of 2.

The problem was generalised from single-link acknowledgements to the case of control message flow in multicast transmission trees by Khanna, Naor and Raz in [16]. Here each recipient of a multicast transmission needs to send an ACK message back to the source of the transmission, i.e., the root of the multicast tree. To reduce control message overhead, ACK's can be aggregated at intermediate nodes, but at the same time their delivery to the root should not be delayed excessively, lest the root reinitiate the multicast transmissions.

Khanna et al. [16] presented a rent-to-buy type balancing heuristic for this problem, and proved that it is $\mathcal{O}(\log C_T)$-competitive, where C_T is the total link cost of the tree. Note that we consider the *centralised* version of the data aggregation problem, where some entity decides which messages to send at each timestep. For its *distributed* version, Khanna et al. established a competitive ratio of $\mathcal{O}(h \log C_T)$, where h is the height of the multicast tree. The latter bound was further improved by Brito, Koutsoupias and Vaya [9] to $\mathcal{O}(C_{\text{maxp}})$, where C_{maxp} is the maximum distance of any node to the root. Similar results were also proved by Oswald, Schmid and Wattenhofer [23]. Korteweg et al. [18] present bicriteria competitiveness bounds on distributed algorithms.

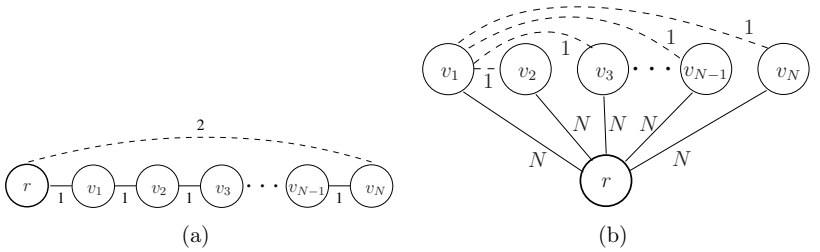

Figure 1: Two networks where simply replacing the graph by a subgraph that is a tree leads to inefficient data aggregation solutions. In network 1(a), the minimum spanning tree does not contain the edge (r, v_N). Thus transmitting a single message from node v_N to the root r has cost N, while the optimal cost is 2. In network 1(b), transmitting simultaneously arriving messages from nodes $v_1 \dots v_N$ via shortest paths to the root r incurs cost N^2, while the optimal cost is only $2N - 1$. In both cases the optimal cost is exceeded by a factor of $\Theta(N)$.

The relevance of data aggregation is, however, not limited to control message management in Internet-like networks. The issue is of central interest particularly in wireless sensor networks [19], where sensors that provide measurements from partly overlapping domains should aim to aggregate their data in order to save in transmission energy costs. Similar problems also appear in the operations research literature in the context of inventory control and lot sizing of logistical networks [2, 10, 20, 21]. Essentially the same model as introduced by Khanna et al. [16] was even evoked independently by Papadimitriou and Servan-Schreiber [24] to explore the structuring of communications in hierarchical human organisations.

To the best of our knowledge, the approximate data aggregation heuristics have so far not been extended beyond treelike networks, even though in many natural applications (sensor networks, logistics, perhaps even organisations) the pertinent networks are not necessarily trees. In fact, Brito et al. [9, p. 634] state that "resolving the competitive ratio of the [task] on arbitrary topologies is an outstanding open problem". The simple approaches of relying on a minimum spanning tree or fixed routing along shortest paths can lead to arbitrarily bad outcomes, as illustrated in Fig. 1.

In this paper, we present a general online algorithm scheme for the data-aggregation problem that is $\mathcal{O}(\log C_{\mathrm{MST}})$-competitive e.g. when the network's ambient graph is of bounded treewidth, or when the graph is *outerterminal*,

by which we mean that one is given a planar embedding of the graph and messages only arrive at its boundary nodes. Our algorithm scheme, described in Section 3, is based on the observation that the rent-to-buy algorithm of Khanna et al. [16] can be generalised to any family of networks, provided one has access to an oracle for the Prize Collecting Steiner Tree (PCST) problem [15] on this family. The prizes in the PCST instance arise from the accumulated delay costs of the messages residing at each node. Since we aim for online algorithms with polynomial time complexity, we only consider efficient algorithms for solving this subproblem.

In Section 5 we address the offline version of the problem in treelike networks, which is a generalisation of the well-known NP-complete joint-replenishment problem [2]. We show that in this setting the approximation ratio of $\mathcal{O}(C_{\mathrm{maxp}})$ established by Brito et al. [9] (cf. also [23]) can be improved by an exponential factor to $\mathcal{O}(\log C_{\mathrm{maxp}})$. This is done by establishing a novel connection between the offline data aggregation problem and a version of the facility location problem with hierarchical opening costs [25].

In an Appendix to this paper, we show how the Dreyfus-Wagner dynamic programming scheme [14] can be applied to obtain an exact PCST algorithm for planar graphs in which nonzero prizes lie only at the outer boundary nodes. This verifies the applicability of the data aggregation algorithm for the corresponding networks, but also serves as a standalone result.

2. Background

Let $G = (V, E, r)$ be a connected (undirected) graph with a specified *root* node $r \in V$. In the data aggregation problem, a set of *messages* $M = \{1, 2, \ldots, N\}$ arrives at the nodes of G over time. Each message needs to be eventually routed to the root via some path originating at its arrival node. Message m accrues *delay cost* at a uniform rate $d > 0$ from its arrival time $t_m \in \mathbb{N}$ until the time of reaching the root; without loss of generality we set $d = 1$. Hence, waiting for t time units causes delay cost t for the message. A transmission of a batch of messages over an edge $e \in E$ gives rise to a fixed *transmission cost* $c(e) \geq 0$, which is independent of the number of messages in the batch. The objective is to find a transmission schedule that forwards all the messages to r while minimising the sum of transmission and delay costs.

We distinguish between *online* and *offline* versions of the problem. In the offline setting, the algorithm has full knowledge of all messages and their

arrival times. In the online setting, on the other hand, the algorithm must make the decisions at time t being aware of only the messages whose arrival times are at or before t. These decisions cannot be revised later. In the online setting we assume, similarly to e.g. [16, 23], that each message incurs a *base cost* of $d = 1$ in any feasible solution, in particular in an optimal solution.

Note that there always exists an optimal solution that has messages wait *only* at the nodes at which they arrived. In this solution, messages are forwarded from their arrival nodes directly to the root in *broadcasts*.[1] Each broadcast is determined by its *broadcast tree* $B = (V_B, E_B)$, and *clears* all nodes in V_B of messages waiting there at the time of the broadcast. In fact, any data-aggregation protocol can be improved (or at least not made worse) by grouping the message transmissions into broadcasts, and so we only consider algorithms of this type. Each message $m \in M$ is then *assigned to* exactly one broadcast that *sends* m after or at time t_m. We denote the set of messages that are assigned to broadcast B by M_B. The *broadcast cost* $C(E_B)$ incurred by B is defined as the total edge cost of the broadcast tree B; this corresponds to the total transmission cost of forwarding the messages cleared by B to the root, taking into account the benefits of data aggregation achieved along the way. More generally, by $C(E')$ for $E' \subseteq E$ we denote the total cost of an edge set E'.

By C_{MST} we denote the cost of a *minimum spanning tree* of the graph G and by C_{maxp} we refer to the maximum distance from any node to the root as determined by the edge costs, i.e., the *maximum shortest path cost* between the root and a node in G. We also use the following notation: $\delta(v, t)$ is the cumulative delay of messages waiting at node v at time t, and $\delta(U, t) = \sum_{v \in U} \delta(v, t)$.

Let $G = (V, E)$ be a connected undirected graph, $c : E \to \mathbb{Q}^+$ an edge cost function, and $V' \subseteq V$ a set of *terminal nodes* in G. The *Minimum Cost Steiner Tree* problem asks for the minimum-cost tree in G that spans all the nodes in V'. In the *Prize-Collecting Steiner Tree (PCST)* problem [3, 15], the fixed set of terminal nodes is replaced by a prize function $p : V \to \mathbb{Q}^+$, and the goal is to find a connected subgraph (V', E') that minimises the sum

$$\sum_{e \in E'} c(e) + \sum_{v \in V \setminus V'} p(v) \quad = \quad C(E') + p(V \setminus V'). \tag{1}$$

[1] Technically, "convergecasts".

Note that in this formulation of the problem, one is not trying to maximise the value of prizes collected from nodes in V', but rather minimise the amount of penalties ordained by *not* selecting nodes in $V \setminus V'$. In general graphs, both the Minimum Cost Steiner Tree and the PCST problems are NP-complete, and also APX-complete to approximate [5]. However, interesting and nontrivial polynomial-time exact and approximate algorithms exist in special types of graphs: in our case most pertinently, the problems can be solved exactly in bounded-treewidth graphs [17], and have PTASs on planar graphs [3, 8]. In Appendix A, we present a polynomial-time dynamic programming algorithm for PCST in graphs that have a planar embedding wherein all nodes with non-zero prizes lie at the outer boundary. For want of a better term, we call such graphs *outerterminal*.

Our data-aggregation algorithm, discussed in Section 3 below, repeatedly selects subsets of nodes that are able to compensate for their broadcast cost by the accumulated delay of messages waiting in them. This task can be formulated as finding a *rooted* PCST in the respective communication network, i.e., a PCST that connects all the selected nodes to the specified root node r. The rooted and unrooted PCST problems are reducible to each other [1], and we only consider the rooted version for the rest of this paper.

3. Online data aggregation in general graphs

A pseudocode version of our general online algorithm scheme BBST (for "Balance and Broadcast Steiner Tree") for the data aggregation problem is presented as Algorithm 1. The scheme is a generalisation of the tree network algorithm of Khanna et al. [16], and follows the rent-to-buy design principle by trying to clear a subset of nodes S by a broadcast over edges E' spanning those nodes, as soon as the cumulative delay cost $\delta(S, t)$ matches the broadcast cost $C(E')$. The crux of the algorithm is that at each time t, it needs to identify the set of nodes V' that maximises the quantity $\delta(V', t) - C(E')$. In general, this would seem to entail a search over an exponential number of sets. In the following we show how to overcome the difficulty of this subproblem in specific types of graphs.

Maximising of the difference

$$\delta(V', t) - C(E') = \delta(V, t) - \delta(V \setminus V', t) - C(E')$$

is equivalent to minimising the sum

$$\delta(V \setminus V', t) + C(E'), \tag{2}$$

```
1  for each timestep t do
2      for each node v ∈ V do
3          update δ(v, t);
4      end
5      S_t ← ∅; E_t ← ∅;
6      while true do
7          (S*, E*) ← arg max_{(S,E') | {r}⊆S⊆V, E' spans S in G} [δ(S, t) − C(E')] ;
8          if δ(S*, t) − C(E*) > 0 then
9              set δ(v, t) ← 0 for v ∈ S*;
10             S_t ← S_t ∪ S*; E_t ← E_t ∪ E*;
11         else
12             exit while;
13         end
14     end
15     if S_t ≠ ∅ then
16         broadcast S_t over E_t;
17     end
18 end
```

Algorithm 1: Balance and Broadcast Steiner Tree (BBST)

which can be seen to be an instance of the PCST problem (1). Let us then assume access to a polynomial-time $(1+\epsilon)$-approximation algorithm for the PCST problem for some $\epsilon \geq 0$. As the minimum value of Equation (2) is upper bounded by $\delta(\emptyset, t) + C_{\mathrm{MST}} = C_{\mathrm{MST}}$, any $(1+\epsilon)$-approximation algorithm has actually an absolute error not greater than $\epsilon\, C_{\mathrm{MST}}$. Now observe that for any subtree (U, F) containing the root, the absolute error condition

$$\delta(V \setminus U, t) + C(F) \leq \min_{\{r\} \subseteq S \subseteq V, E' \text{ spans } S \text{ in } G} [\delta(V \setminus S, t) + C(E')] + \epsilon\, C_{\mathrm{MST}}$$

implies also

$$\max_{(V',E')} [\delta(V', t) - C(E')] - \epsilon\, C_{\mathrm{MST}} \leq \delta(U, t) - C(F) \leq \max_{(V',E')} [\delta(V', t) - C(E')]\,. \tag{3}$$

The details of Algorithm 1 are written so as to tolerate such approximate solutions in place of exact solutions at line 7. For the case of an exact PCST solver ($\epsilon = 0$) the operations can be simplified. Also we remark that equality of the condition at line 8 is not allowed in order to prevent empty sets from being handled as broadcasts. Hence each iteration of the while-loop at lines 6 thru 14 either adds at least one new node to the broadcast set or finishes the loop, and the algorithm is thus guaranteed to terminate.

3.1. Proof of BBST competitiveness

The competitiveness proof for algorithm BBST follows in broad outline that of Khanna et al. [16]. First we need some auxiliary lemmas.

Lemma 1. *In BBST, total broadcast costs are upper bounded by total delay costs.*

Proof. We prove the lemma separately for each broadcast of BBST, implying the stated result.

For each t the broadcast node set S_t is constructed incrementally from parts that have individually no more broadcast than delay costs. Since the broadcast costs are subadditive and delay costs are additive, broadcast costs stay below the delay costs after every increment. □

Let $\delta(m)$ for $m \in M$ denote the delay cost imposed by algorithm BBST on message m. Similarly, define $\delta(M') = \sum_{m \in M'} \delta(m)$ for any subset $M' \subseteq M$. For an optimal broadcast sequence OPT, denote the corresponding terms as $\delta^*(m)$ and $\delta^*(M')$.

Partition the set of messages M according to the broadcasts in an optimal sequence OPT, and fix a broadcast in OPT occurring at time t. Let $M_t \subseteq M$ be the corresponding set of messages, and denote the total cost of the broadcast tree used by OPT at time t by C_t. Divide the set of messages in M_t further into two subsets $M_t^-, M_t^+ \subseteq M_t$ according to whether they are broadcast by algorithm BBST earlier (M_t^-) or later (M_t^+) than at time t.

Clearly, the messages in M_t^- cannot cause more delay cost for BBST than for OPT, i.e., we have

$$\delta(M_t^-) \leq \delta^*(M_t^-) \leq \delta^*(M_t).$$

Let us then consider the delay cost due to messages in M_t^+ over the time interval (t, ∞). The delay cost accumulated during interval $(t, t+1]$ is bounded simply by $|M_t^+|$, so consider the interval $(t+1, \infty)$.

Lemma 2. *Let $r \in \mathbb{Z}_+$. The number of messages in M_t^+ that are still waiting after time $t + r$ is at most*

$$\frac{C_t + \epsilon\, C_{MST}}{r}.$$

Proof. Suppose the contrary. Then the total delay cost at time $t + r$ incurred by the messages in M_t^+ that are still waiting after time $t + r$ is strictly greater than

$$\frac{C_t + \epsilon\, C_{\mathrm{MST}}}{r} \cdot r = C_t + \epsilon\, C_{\mathrm{MST}}.$$

Let U be the set of nodes that contain waiting messages in M_t^+, and let F be the set of edges in the OPT broadcast tree at time t. We obtain the bound

$$\begin{aligned} \max_{(V',E')} [\delta(V', t+r) - C(E')] &\geq \delta(U, t+r) - C(F) \\ &> C_t + \epsilon\, C_{\mathrm{MST}} - C_t = \epsilon\, C_{\mathrm{MST}}, \end{aligned}$$

Based on equation (3), one thus concludes that $\delta(S^*, t+r) - C(E^*) > 0$ for every (S^*, E^*) pair returned by the PCST subroutine run at time $t + r$ on line 7 of algorithm BBST. But this contradicts the eventual termination of the main while loop of lines 6–14 at time $t + r$. From this contradiction we conclude that the claim of the Lemma must be true. □

Corollary 3. *For any given OPT broadcast time t, after time $t+(C_t+\epsilon\, C_{MST})$ there are no more messages from M_t^+ waiting in algorithm BBST.*

Proof. Apply Lemma 2 with $r = C_t + \epsilon\, C_{\mathrm{MST}} + \delta$ for arbitrarily small $\delta > 0$. □

Now we are ready to bound the total delay cost paid by algorithm BBST.

Lemma 4.

$$\delta(M) = \mathcal{O}\left((1+\epsilon\, C_{MST}) \ln((1+\epsilon) C_{MST})\right) OPT.$$

Proof. Consider any broadcast in the sequence OPT, effected at some time t. Split the corresponding delay period for algorithm BBST into two intervals $[t, t+1]$ and $(t+1, C_t + \epsilon\, C_{\mathrm{MST}}]$ The total delay before $t+1$ is bounded by $|M_t^+|$, and using Lemma 2 for the latter interval, we conclude the following upper bound for BBST delay costs:

$$\begin{aligned}\delta(M_t^+) &\le |M_t^+| + \int_1^{(C_t + \epsilon\, C_{\mathrm{MST}})} \frac{C_t + \epsilon\, C_{\mathrm{MST}}}{r} \cdot dr \\ &\le \delta^*(M_t) + (C_t + \epsilon\, C_{\mathrm{MST}})\ \ln\left(C_t + \epsilon\, C_{\mathrm{MST}}\right).\end{aligned}$$

Furthermore,

$$\begin{aligned}\delta(M_t) &= \delta(M_t^-) + \delta(M_t^+) \\ &\le 2 \cdot \delta^*(M_t) + (C_t + \epsilon\, C_{\mathrm{MST}})\ \ln((C_t + \epsilon\, C_{\mathrm{MST}})) \\ &\le 2 \cdot \delta^*(M_t) + C_t\ \ln((1+\epsilon) C_{\mathrm{MST}}) + \epsilon\, C_{\mathrm{MST}}\ \ln((1+\epsilon) C_{\mathrm{MST}}).\end{aligned}$$

Summing over all broadcasts $\mathcal{B}$ in the sequence OPT yields

$$\begin{aligned}\delta(M) &\le (2 + \ln((1+\epsilon) C_{\mathrm{MST}}))\,\mathrm{OPT} + |\mathcal{B}| \epsilon\, C_{\mathrm{MST}}\ \ln((1+\epsilon) C_{\mathrm{MST}}) \\ &\le \left[2 + (1+\epsilon\, C_{\mathrm{MST}}) \ln((1+\epsilon) C_{\mathrm{MST}})\right] \mathrm{OPT},\end{aligned}$$

as every broadcast must have at least one message with base cost 1. The claim follows. □

Recall from Lemma 1 that the broadcast costs paid by algorithm BBST are dominated by the messages' delay costs. As a base cost of 1 unit per message needs to be paid by any possible broadcast sequence, the delay cost bound given in Lemma 4 yields the desired result on the competitive ratio of algorithm BBST.

Theorem 5.

$$\frac{BBST}{OPT} = \mathcal{O}\left((1+\epsilon\, C_{MST}) \ln((1+\epsilon) C_{MST})\right).$$

4. Online data aggregation in special graph classes

Applying the BBST algorithm scheme of Section 3 to a graph class $\mathcal{G}$, where the ancillary PCST problem can be solved efficiently to precision $\epsilon \geq 0$, yields an online algorithm for the data aggregation problem on $\mathcal{G}$, with a competitive ratio given by Theorem 5. We consider first the special case of graphs with bounded treewidth.

A *tree decomposition* of a connected graph $G = (V, E)$ is a tree $\mathcal{T} = (\{V_1, \ldots, V_N\}, E_{\mathcal{T}})$, where each $V_i \subseteq V$ and $\cup_i V_i = V$. For every edge $\{u, v\} \in E$ there must be an i such that $\{u, v\} \subseteq V_i$, and for every $v \in V$ the subgraph of $\mathcal{T}$ induced by the set $\{V_i \mid v \in V_i\}$ has to be connected. The *width* the decomposition $\mathcal{T}$ is $\max_i |V_i| - 1$, and the *treewidth* of G is the minimum width of any tree decomposition of G. For instance, series-parallel graphs and outerplanar graphs have treewidth at most 2, and *Halin graphs* have treewidth at most 3. (Cf. e.g. [6].)

Many in general NP-complete graph optimisation problems can be solved in polynomial time on graphs of constant-bounded treewidth, by a dynamic programming scheme following the tree decomposition structure [7]. In particular, for minimum-cost Steiner trees on bounded-treewidth graphs, Korach and Solel [17] presented a linear-time algorithm which in high-level description operates as follows:

1. Consider the nodes V_i in a tree decomposition $\mathcal{T}$ of a graph G in order from leaves towards the root.
2. For each decomposition node V_i, consider all possible induced forests that an optimal Steiner tree might produce, when restricted to the nodes of G contained in V_i.
3. For a given decomposition node V_i, consider all of its children in $\mathcal{T}$ in order, and for each child node V_j merge its induced forests to those already associated to V_i in all possible ways, pruning out combinations that are seen to be suboptimal or infeasible.

By including the prize values of selected terminals in the induced subforests the Korach-Solel algorithm can be extended to compute also minimum-cost PCSTs. Also in a recent paper [11], a similar algorithm is presented explicitly for the PCST problem. Thus, by setting $\epsilon = 0$ in Theorem 5 we obtain the following corollary:

Corollary 6. *For any family of bounded-treewidth networks, there is a polynomial-time online algorithm for the data aggregation problem, with a*

competitive ratio of $\mathcal{O}(\log C_{MST})$, where C_{MST} is the cost of the minimum spanning tree of the network under consideration.

A second class of graphs for which the PCST problem can be solved exactly in polynomial time are planar graphs with nonzero prizes only on the boundary, which we call *outerterminal* graphs. An efficient Steiner tree algorithm was given in [4] for the somewhat more general class of planar graphs where the terminals lie within some constant number of layers away from the boundary. The extension of this approach to PCSTs in outerterminal graphs is worked out in Appendix A.

Corollary 7. *For any family of outerterminal networks, there is a polynomial-time online algorithm for the data aggregation problem, with a competitive ratio of $\mathcal{O}(\log C_{MST})$, where C_{MST} is the cost of the minimum spanning tree of the network under consideration.*

An exact PCST algorithm is not always required for meaningful results. Whenever the $(1+\epsilon)$-approximation for PCST is available as an FPTAS, i.e., the time complexity is polynomial in $\frac{1}{\epsilon}$, one may use $\epsilon = \frac{1}{C_{\text{MST}}}$ and hence have BBST running time depend polynomially on C_{MST}. Since the dependence is on the *value* of C_{MST} rather than the size of its bit-encoding, the algorithm is strictly speaking only *pseudopolynomial-time*.

Corollary 8. *For any family of networks where the* PCST *problem admits an FPTAS, there is a pseudopolynomial-time online algorithm for the data aggregation problem, with a competitive ratio of $\mathcal{O}(\log C_{MST})$, where C_{MST} is the cost of the minimum spanning tree of the network under consideration.*

There are several graph classes that are good candidates of admitting at least an FPTAS for the PCST problem and thus falling under Corollary 8. For example, Steiner tree PTASs are known for c-local networks [26] and unit disk graphs [22]. Also the recent PTAS for planar graphs [3] might extend to an FPTAS for some subclasses of planar graphs.

5. Offline data aggregation in trees

We now consider the offline version of the problem in trees and present an approximation algorithm based on a relaxation of the problem. Note that since zero-cost edges are allowed, without loss of generality we can assume that messages only arrive at leaf nodes.

5.1. Problem relaxation

The data aggregation problem as defined in Section 2 penalises delay and of course forbids completely broadcasting messages before their arrival times. In the relaxed problem we allow a message to be "sent" before its arrival time and replace its delay cost in the objective function by the difference between the broadcast and arrival times. We call this version a *two-way* data aggregation problem and use the term *one-way* to refer to the original problem.

The two-way problem can be considered an instance of the facility location problem with *hierarchical facility opening costs*, which is approximable to a constant factor by the algorithm due to Svitkina and Tardos [25]. In the facility location problem one is given a set of clients and candidate facility locations. The task is to compute an assignment of clients to open facilities that minimises the sum of assignment and facility opening costs. Depending on the structure of assignment and opening costs, various cases are possible. The algorithm proposed in [25] was formulated for metric assignments costs and opening costs that depend on the set of clients assigned to a facility.

It is easy to see that for the data aggregation problem it suffices to consider only message arrival times as candidates for broadcast times. Hence, the two-way model of our problem lends itself to a reduction to the facility location problem, where the clients are messages and facilities are broadcasts at arrival times of messages. Each facility has the same hierarchical opening cost function, which corresponds to the cost of the subtree connecting the leafs of the assigned messages to the root. The assignment costs are determined by the distance in time between arrival and broadcast time.

5.2. Algorithm and analysis

The approximation algorithm operates in three steps. The first step relaxes the problem and constructs a two-way solution via the algorithm in [25]. The two later steps transform the solution back into a feasible one-way solution, where the second step applies local improvements: whenever the total cost can be decreased by broadcasting a message on its own, its is sent at its arrival time.

The third-step subroutine Reassign is presented in detail as Algorithm 2, which is called for each broadcast with messages being sent before their arrival times. Its operation is visualised in Fig. 2. The continuous arrows correspond to message assignments before running Reassign, while the dashed arrows and dashed lines correspond to new assignments and broadcasts after

```
1  function Reassign [B]                    // B is the original broadcast
2      t_B ← time of broadcast B;
3      M_B ← set of messages assigned to B;
4      t_f ← time of last arrival of messages in M_B;
5      while t_f > t_B do
6          insert a new broadcast B' at time t_f;
7          for each m ∈ M_B arriving at or after (t_B + t_f)/2 do
8              M_B' ← M_B' ∪ {m};
9              M_B ← M_B \ {m};
10         end
11         t_f ← time of last arrival of messages in M_B;
12     end
13 end
```

Algorithm 2: Reassign

the run. Reassign repeatedly reduces the number of messages sent too early by assigning the rightmost messages to a new appropriately timed broadcast.

Theorem 9. *The solution thus constructed within a factor of* $\mathcal{O}(\log(C_{maxp}))$ *of an optimal solution for the data aggregation problem.*

For a proof we need some auxiliary results. Notably, we show that the second and third steps of the main algorithm constitute a subroutine that transforms any two-way feasible solution into a feasible one-way solution, where the new cost is within a factor $\log(C_{\text{maxp}}) + 1$ of the original.

Consider the second step of the algorithm. By definition it can only decrease total cost. After termination of the step the maximum single-message broadcast cost C_{maxp} is an upper bound for the delay or earliness costs of a message, and hence an upper bound for the difference between message arrival and broadcast times.

Lemma 10. *After the second step, the waiting or earliness time of a message is at most* C_{maxp}.

Now consider the third step and a broadcast B, whose broadcast time t_B is before the arrival time t_f of the latest message assigned to B. The message set M_B assigned to B is going to change during the process, so let us denote

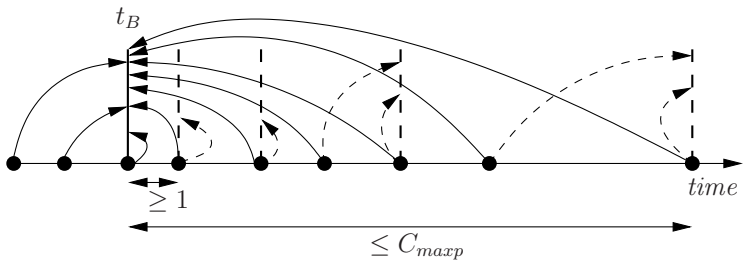

Figure 2: Procedure Reassign transforming a two-way solution into a one-way solution. The solid arrows correspond to message assignments before running Reassign, while the dashed arrows and dashed lines correspond to new assignments and broadcasts after the run.

by $C_2(B)$ the broadcast cost $C(B)$ with the message set before running third step. We prove an upper bound for the number of loops in the third step for the broadcast, and thereafter an upper bound to the cost increase for running the loop once.

Lemma 11. *For a given B, the Reassign algorithm loop runs at most $\log_2(C_{maxp})$ times.*

Proof. For each iteration, the time span between broadcast time t_B and arrival time t_f of last message assigned to B gets halved or reduced even more. Lemma 10 bounds the maximum initial span, whence iterating at most $\log(C_{\text{maxp}})$ times is enough to reach $t_f - t_B < 1$. Because t_B is a message arrival time, and the model assumed integer arrivals, $t_f - t_B < 1$ actually implies $t_f \leq t_B$. Thus Reassign terminates in at most $\log(C_{\text{maxp}})$ iterations. □

Lemma 12. *For a given B, each run of the loop in Reassign increases the total cost of the two-way solution by at most $C_2(B)$.*

Proof. Each of the messages reassigned from B to B' was previously sent at least $(t_f - t_B)/2$ time units early, and is now sent at most $(t_f - t_B)/2$ units late. Earliness cost is thus transformed into smaller or equal delay cost, and the only cost increase can be due to the new broadcast costs $C(B') \leq C(B) \leq C_2(B)$. □

Note that each newly introduced broadcast B' in third step sends messages only after their arrival times. Thus, the two-way solution is transformed into a solution that sends no messages before their arrival times, making it a feasible one-way solution. We conclude the following lemma:

Lemma 13. *Steps 2 and 3 of the Reassign algorithm transform any two-way feasible solution into a feasible one-way solution, whose cost is within a factor* $1 + \log(C_{maxp})$ *of the original.*

Now we are ready to prove the main result.

Proof of Theorem 9. Let the optimal costs for the one-way and two-way problems have values OPT_1 and OPT_2, respectively. Let ALG_2 be the cost of the two-way solution obtained in step 1, and ALG_1 cost of the final one-way solution.

At first, Lemma 13 shows that we can bound the final cost as $\mathrm{ALG}_1 \leq (1 + \log C_{\mathrm{maxp}})\,\mathrm{ALG}_2$. From the result of the facility location algorithm in [25] it follows that $\mathrm{ALG}_2 \leq 4.24\,\mathrm{OPT}_2$. As any one-way solution is also a two-way solution, $\mathrm{OPT}_2 \leq \mathrm{OPT}_1$ is immediate. Combining these results yields the final claim as

$$\mathrm{ALG}_1 \leq 4.24\,(1 + \log C_{\mathrm{maxp}})\,\mathrm{OPT}_1.$$

□

6. Conclusions

We considered the data aggregation problem that arises naturally in many varieties of communication networks, such as sensor networks, where one can trade off the benefits of aggregating messages against the penalty of message delays. Many versions of this problem on treelike networks have been considered previously, but to the best of our knowledge, we present the first results in networks different from trees. Further, the only previously known offline data aggregation algorithm improving over online results is for the special case of the joint replenishment problem, i.e., trees of height two, where the offline 1.8-approximation [21] improves over online 3-approximation [10].

An issue of interest concerns improving the competitive ratios achieved. We have basically extended the competitive ratio $\mathcal{O}(\log C_{\mathrm{MST}})$ obtained by Khanna et al. from trees to other network families. Khanna et al. ask whether there exists an online data aggregation algorithm on trees with a constant competitive ratio, and it is quite natural to ask the same question for instance

of the bounded-treewidth or outerterminal graphs. From e.g. the sensor networks applications point of view, an online algorithm with a nontrivial competitive ratio for all planar networks would also be of notable interest.

Acknowledgements

The first and second author have been supported by Academy of Finland under grants 128823 and 1313660, respectively. The first author has also been supported by the Helsinki Doctoral Programme in Computer Science and the Nokia Foundation. The third author was supported by the Cloud Software Programme of the Finnish Strategic Centre for Science, Technology and Innovation TiViT. Part of this work was done while the first and second author were visiting the Institute of Computer Science, Humboldt-Universität zu Berlin.

Appendix A. Polynomial-time solvability of PCST in outerterminal graphs

In this appendix we show that the PCST problem, introduced in Section 2, is solvable in polynomial time in outerterminal graphs. Recall that by outerterminal we refer to the property of being planar and the additional condition that all nodes that have positive prizes lie on the boundary of the graph for some fixed planar embedding, which is given to the algorithm. Also the graph has a specific root node r.

For convenience, instead of Eq. (1), we consider the equivalent objective of minimising the cost of edges used minus the prizes collected

$$\sum_{e \in E'} c(e) - \sum_{v \in V'} p(v). \tag{A.1}$$

Here V' and E' are the nodes and edges of the tree. For a given tree $T = (V', E')$ we refer to the value of Eq. (1) that it achieves as the *cost* of the tree T. Without loss of generality we may assume $p(r) = 0$ as the root must be included in any tree and its prize can be collected at no additional cost. In this convention, a trivial tree containing only the root node has zero cost, and an optimal tree has nonpositive cost.

In solving the rooted PCST we can w.l.o.g. assume that the graph contains only edges with strictly positive cost. Any minimum-cost subgraph containing r must then be a tree, which simplifies the following presentation. Additionally we assume that the graph boundary is *clean* in the sense that there exists a simple path traversing all boundary edges exactly once. This assumption is not restrictive, as it can be overcome by a separate dynamic program that splits the graph at each boundary node v with multiple visits and solves the PCST problem in the splitted subgraphs. However, for simplicity we focus directly on the clean boundary case.

High-level idea

We obtain our algorithm for PCST on outerterminal graphs by a modification of the dynamic programming algorithm due to Dreyfus and Wagner [14] (as presented in [4]). The original algorithm has already resulted in polynomial time algorithms, by Bern and Bienstock [4], for the Steiner tree problem in special cases of planar graphs.

The recursion of the original Steiner tree algorithm is based on the observation that for any node v that is part of an optimal Steiner tree T^*, one of the following must hold:

(a) v has degree 2 or larger in T^*; in this case the solution T^* decomposes (*splits*) into two optimal solutions, which each connect v to a non-empty subset of the terminal nodes.

(b) v has degree 1 in T^* (and thus is a terminal); in this case T^* connects v via a shortest path to another node u, which either is a non-terminal node (and thus must have degree at least 2), or to a different terminal $u \neq v$.

We extend the recursion in the Bern-Bienstock algorithm [4] by introducing prizes located at the terminal nodes and relaxing the constraint that all terminals must be connected to a given root, thus obtaining a PCST algorithm. As also observed in [4] for the Steiner tree problem, when terminals are located at the outer boundary of a planar embedding, for case (a) it suffices to only consider splits of terminals that result from *intervals* of consecutive boundary nodes according to some (clockwise or anti-clockwise) ordering. This ensures polynomial runtime as there is only a polynomial number of boundary intervals, even if the total number of boundary subsets is exponential.

Definitions

We begin by stating preliminary definitions that are used in the PCST algorithm or its analysis. Let V_B be the set of boundary nodes of the given clean planar embedding of graph G, and V_I the set of interior nodes, so that $G = (V_I \cup V_B, E)$. Denote the set of boundary edges by E_B. An ordered set $L = \{l_1, l_2, \dots l_k\} \subseteq B$ is an *interval* if $l_1 l_2 \dots l_k$ is a simple path counterclockwise along the boundary. Denote by $E(L)$ the set of edges between the consecutive nodes in L,

$$E(\{l_1, l_2, \cdots, l_k\}) = \{(l_1, l_2), \cdots, (l_{k-1}, l_k)\}.$$

If L' and L are intervals such that $E(L') \subseteq E(L)$, we say that L' is a *subinterval* of L. We write $L' \sqsubset L$ when both L and $L \setminus L'$ are nonempty subintervals of L.

When $u \in L$, we define L_u^- as the longest subinterval of L that has an end point at u's clockwise boundary neighbour. Note that L_u^- is empty

if the clockwise neighbour of u is not in L. L_u^+ is defined as the interval $L \setminus (\{u\} \cup L_u^-)$. When $u, u' \in L$, $u \neq u'$ and $u' \in L_u^-$, we say that that u *lies after* u' *on* L.

Let $\mathrm{rot}(G)$ be the set of all *rotations* of V_B, i.e., the set of all intervals L with $|L| = |V_B|$ that can be constructed by choosing a different start node from V_B. Note that $|\mathrm{rot}(G)| = |V_B|$.

For $v \in V_I \cup V_B$, a v-rooted tree T and a node $v' \in T$, denote by $T_{v'}$ the subtree of T rooted at v', which contains exactly the nodes u such that the path between v and u in T contains v'. The remaining subtree is denoted by $T \setminus T_{v'}$, and it contains exactly the nodes that are reachable from v after v' is removed from T.

Consider the planar embedding of G to $\mathbb{R}^2$. For an r-rooted tree T in G define *escape curves* as follows. An escape curve $\gamma : [0, 1] \mapsto \mathbb{R}^2$ is a continuous mapping to the plane satisfying the following:

- $\gamma(0) = r$,
- $\gamma(1)$ lies on the infinite face, i.e., it lies outside the graph boundary E_B,
- for $s > 0$, γ does not intersect with edges of T or any nodes $(V_B \cup V_I)$, and
- γ crosses over E_B exactly once.

Observe that if r itself lies on the boundary V_B, since any valid escape curve emanates from r, it also intersects with the interior before it leaves via any of the boundary edges, possibly via one of the two incident to r. An edge $e \in E_B \setminus T$ is an *escape edge* $e \in \mathrm{esc}(T)$ if there is an escape curve γ that crosses E_B at e. An escape curve is illustrated in Fig. A.3.

Let L be an interval on the boundary nodes. We say that a rooted tree T is L-*consistent* if there exists an $e \in \mathrm{esc}(T) \setminus E(L)$, i.e., there exists an escape edge outside interval L.

Lemma A.1. *Let T be a v-rooted tree that is L-consistent, L' a subinterval of L, $v' \neq v$ a node in T, and v_c a child of v. Then:*

1. *T is L'-consistent,*
2. *Any v-rooted subtree T' of T is L-consistent,*
3. *If $v' \notin L$, $T_{v'}$ is L-consistent,*
4. *If $v' \in L$, $T_{v'}$ is both $L_{v'}^+$-consistent and $L_{v'}^-$-consistent.*

Proof.

1. Since $L' \subseteq L$ implies $E(L') \subseteq E(L)$ and $\mathrm{esc}(T) \setminus E(L) \subseteq \mathrm{esc}(T) \setminus E(L')$, the claim is immediate.
2. The set of edges in T' is a subset of edges in T; hence $\mathrm{esc}(T) \subseteq \mathrm{esc}(T')$ and the result follows.
3. Recall that none of the nodes on the $v' - r$ path are in $T_{v'}$, except for v' itself. An escape curve for $T_{v'}$ can be constructed by following the $v' - r$ path and connecting to the escape curve for T. The escape edge is outside L.
4. The argumentation of the previous case yields an escape edge not in L and hence in neither of $L^-_{v'}$ and $L^+_{v'}$.

□

For an interval L and a tree T with nodes V' and edges E', we define the *L-cost* of T as $\sum_{e \in E'} c(e) - \sum_{v \in L \cap V'} p(v)$. Thus, the L-cost is the cost of the tree when prizes are taken into account only from nodes in L.

We state the following definition.

Definition A.2. *For each $v \notin L$, define $\widehat{C}(v, L)$ as the smallest L-cost of an L-consistent v-rooted tree T.*

Recursion

We now proceed to our efficient PCST algorithm for outerterminal graphs. The core component of the algorithm is a recurrence equation for the optimal PCST cost, restricted to solutions of subproblems that are L-consistent with respect to interval splits. As we shall show below, this restriction suffices to achieve a total cost that is equal to the unconstrained optimal PCST cost for the given instance.

For a boundary interval L and node $v \notin L$, define $C(v, L)$ through

$$\begin{aligned} C(v, L) = \min \Big\{ & \min_{L' \sqsubset L} \min_{u \in V \cup \{v\}} \left(d(v, u) + C(u, L') + C(u, L \setminus L') \right), \\ & \min_{u \in L} \left(d(v, u) - p(u) + C(u, L^-_u) + C(u, L^+_u) \right), \\ & 0 \Big\}. \end{aligned} \tag{A.2}$$

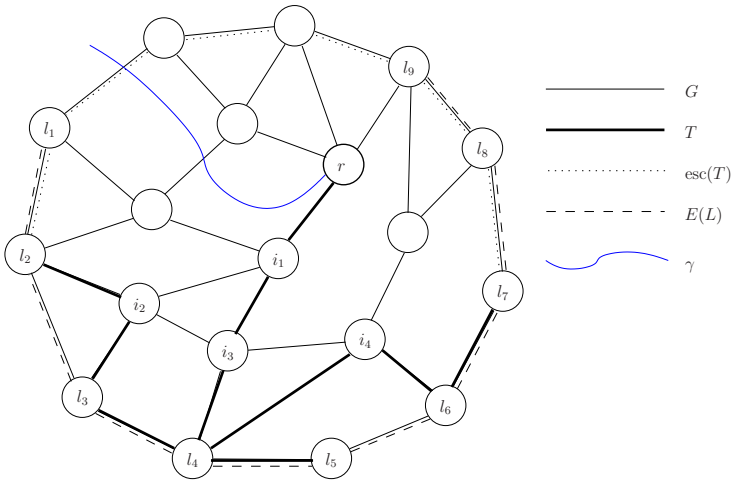

Figure A.3: $u = l_4$, $L = \{l_1 \dots l_9\}$, $L_u^- = \{l_1 \dots l_3\}$, $L_u^+ = \{l_5 \dots l_9\}$. Tree T^- contains nodes l_4, l_3, i_2 and l_2, while T^+ contains nodes l_4, l_5, i_4, l_6 and l_7.

Time complexity: Each evaluation of Equation (A.2) decreases the size of the interval parameter L. Further, observe that when $L = \emptyset$, the recursion evaluates to 0 by definition. As there are $\mathcal{O}(n)$ nodes v and $\mathcal{O}(n^2)$ intervals L, there are $\mathcal{O}(n \cdot n^2) = n^3$ entries of $C(\cdot, \cdot)$ to be computed, which can be done in polynomial time.

Theorem A.3. *The optimal PCST cost in the graph G is*

$$OPT_{PCST} = \min_{L \in rot(G)} C(r, L),$$

and a corresponding tree can be found by backtracking the C minimisation steps.

The proof is divided into three lemmas (A.4)-(A.6). Strictly speaking, the lemmas imply only that an optimal spanning subgraph G' can be found, but optimality and the absence of zero-cost edges guarantee that G' is in fact a tree.

Remark: To deal with the case $r \in L \subseteq V_B$, we interpret $C(r, L) := C(r, L_r^-) + C(r, L_r^+)$. The similar extension is applied to $\widehat{C}$.

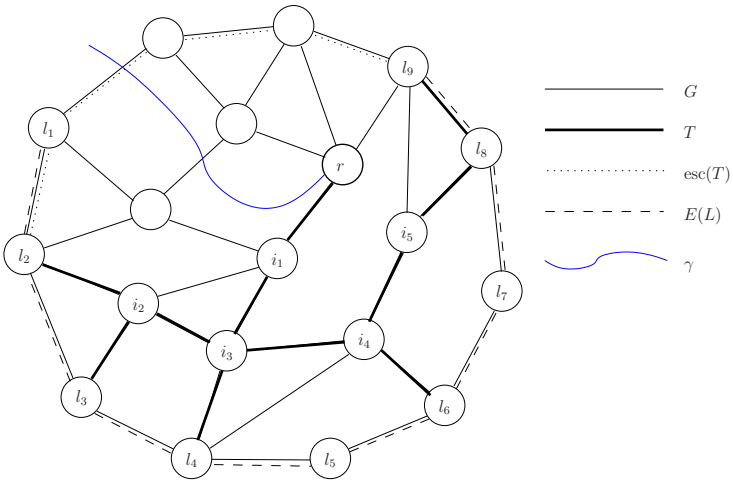

Figure A.4: $u = i_3, u' = i_2, v_1 = l_2, v_2 = l_3$. $L' = \{l_1, l_2, l_3\}$. Tree T_1 contains nodes i_3, i_2, l_2 and l_3, while T_2 contains nodes $i_3, l_4, i_4, l_6, i_5, l_8$ and l_9.

Lemma A.4.

$$OPT_{PCST} = \min_{L \in rot(G)} \widehat{C}(r, L).$$

Proof. Fix an optimal PCST solution T on G. As T has no cycles it cannot enclose r. Hence there exists an escape curve from r to the infinite face that crosses an escape edge e. Choose $L' \in \text{rot}(G)$ so that its beginning and end are endpoints of e, making $e \notin E(L')$. Hence T is L'-consistent, and as L' contains the whole boundary, its L'-cost matches the ordinary cost. These facts imply

$$\widehat{C}(r, L') \leq OPT_{PCST}$$

by definition of $\widehat{C}$. The reverse inequality is immediate. □

Lemma A.5. *For each v, L,*

$$\widehat{C}(v, L) \geq C(v, L).$$

Proof. We give the proof by induction on $|L|$. Fix v. Observe that (A.2) gives $C(v, \emptyset) = 0 = \widehat{C}(v, \emptyset)$, which verifies the base case $|L| = 0$. Then assume

$\widehat{C}(v, L) \geq C(v, L)$ for all $|L| \leq k - 1$. Now fix $|L| = k$ with $v \notin L$ and an L-consistent tree T that is rooted at v and yields the minimum of $\widehat{C}(v, L)$.

We distinguish between different cases of v's degree, that is the outdegree of v in T. Consider first the special case of v having degree 0 in T. Then T only consists of the root v and $\widehat{C}(v, L) = 0 \geq C(v, L)$ due to the assumption that $p(r) = 0$ and the third component in Eq. (A.2).

Next, assume the degree of v in T is one. Following the outbound path from v eventually leads to an L node or a Steiner node. In the case where one reaches first a boundary node $u \in L$, consider T_u. For each of u's children u', $L \cap T_{u'}$ must be contained in either L_u^+ or L_u^-, due to T being L-consistent. Let us split T_u accordingly. Let T^- be the subtree of T_u, rooted at u, cutting off each $T_{u'}$ where u' is a child of u and $T_{u'} \cap L \subseteq L_u^+$. Define T^+ similarly, cutting off $T_{u'}$ with $T_{u'} \cap L \subseteq L_u^-$. We conclude

$$\begin{aligned}
\widehat{C}(v, L) &= d(v, u) - p(u) + T^+ + T^- \\
&\geq d(v, u) - p(u) + \widehat{C}(u, L_u^+) + \widehat{C}(u, L_u^-) \\
&\geq d(v, u) - p(u) + C(u, L_u^+) + C(u, L_u^-) \\
&\geq C(v, L)
\end{aligned}$$

where T^+, T^- refer to the costs of the corresponding trees. The first inequality is due to T^+, T^- being consistent on L_u^+, L_u^- respectively by Lemma A.1.4, and collecting prizes from only L_u^+, L_u^- respectively. The second inequality is the induction assumption, and the third holds directly by definition of C.

The remaining cases to prove are v having degree at least two, or v having degree one with the outbound path leading to a Steiner node u before a boundary node. In such cases we can identify a unique node $u \notin L$, possibly $u = v$, which is closest to v among nodes that have at least two children in T. Observe that T and T_u contain the same set of leaf nodes.

On L, identify the node v_1 of T that lies before any other $T \cap L$ node on L. This node must belong to exactly one $T_{u'}$ where u' is a child of u. Define T_1 as the subtree of T with nodes $T_{u'} \cup \{u\}$. Identify the node v_2 of $T_{u'}$ that lies after any other $T_{u'} \cap L$ node on L, and define $L' \sqsubset L$ as spanning from start of L to v_2. Note that $L \setminus L'$ is nonempty, since we assumed that u has degree at least two in T, which means there is another leaf node outside T_u. By planarity of G the leaf node lies on the boundary outside L'. Since $T_{u'} \cap L \subseteq L'$, the cost of T_1 matches its L'-cost, and T_1 is L'-consistent by Lemma A.1.1 and Lemma A.1.2.

Let us further show that for the tree $T_2 = T \setminus T_{u'}$, its cost matches its $L \setminus L'$-cost and the tree is $(L \setminus L')$-consistent. To prove the first claim, assume $T \setminus T_{u'}$ contains a node w in L. Since T is a tree, it has no cycles and $w \notin T_{u'}$. By definition of v_1, v_1 lies before w on L. Tree $T_{u'}$ connects u' to the boundary nodes v_1 and v_2, so by planarity and noncyclicity of T it must be the case that v_2 lies before w. Hence $w \in L \setminus L'$, and the cost of $T \setminus T_{u'}$ equals its $L \setminus L'$-cost. The $(L \setminus L')$-consistency of $T \setminus T_{u'}$ is immediate from Lemma A.1.2 and Lemma A.1.1.

Hence we can decompose the cost of T as

$$\begin{aligned}\widehat{C}(v, L) &= d(v, u) + T_1 + T_2 \\ &\geq d(v, u) + \widehat{C}(u, L') + \widehat{C}(u, L \setminus L') \\ &\geq d(v, u) + C(u, L') + C(u, L \setminus L') \\ &\geq C(v, L)\end{aligned}$$

where T_1 and T_2 refer to the costs of the corresponding trees.

In any case $\widehat{C}(v, L) \geq C(v, L)$, and the induction is finished. □

Lemma A.6. *For each v, L,*

$$\min_{L \in rot(G)} C(r, L) \geq OPT_{PCST}.$$

Further, the value of $C(v, L)$ always corresponds to a subgraph connecting v to boundary nodes in L, and the tree can be found by backtracking the minimisation steps taken.

Proof. The process in Eq. (A.2) can be seen as accumulating a collection of edges and their costs minus prize values collected to $C(v, L)$. Note that each prize can be included at most once to the accumulated C value, and for each included prize $p(u)$ there is a path from v to u included in edges whose cost is added. The subgraph G' corresponding to $C(v, L)$ contains the root and some (possibly zero) boundary nodes and is connected, so it is a valid solution to the problem of connecting v to the prize nodes in L.

□

References

[1] A. Archer, M. H. Bateni, M. T. Hajiaghayi, and H. Karloff. Improved approximation algorithms for Prize-Collecting Steiner Tree and TSP. In *Proc. 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 427–436, 2009.

[2] E. Arkin, D. Joneja, and R. Roundy. Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters*, 8(2):61–66, 1989.

[3] M. Bateni, C. Chekuri, A. Ene, M. T. Hajiaghayi, N. Korula, and D. Marx. Prize-collecting Steiner problems on planar graphs. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 1028–1049, 2011.

[4] M. Bern and D. Bienstock. Polynomially solvable special cases of the Steiner problem in planar networks. *Annals of Operations Research*, 33:403–418, 1991.

[5] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[6] H. L. Bodlaender. A tourist guide through treewidth. Technical Report RUU-CS-92-12, Department of Computer Science, Utrecht University, March 1993.

[7] H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded trewidth. *Computer Journal*, 51:255–269, 2008.

[8] G. Borradaile, C. Kenyon-Mathieu, and P. Klein. A polynomial-time approximation scheme for Steiner tree in planar graphs. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007)*, pages 1285–1294, 2007.

[9] C. Brito, E. Koutsoupias, and S. Vaya. Competitive analysis of organization networks or multicast acknowledgement: How much to wait? In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, pages 627–635, 2004.

[10] N. Buchbinder, T. Kimbrel, R. Levi, K. Makarychev, and M. Sviridenko. Online make-to-order joint replenishment model: Primal dual competitive algorithms. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*, pages 952–961, 2008.

[11] C. Chekuri, A. Ene, and N. Korula. Prize-collecting Steiner tree and forest in planar graphs. *ArXiv e-prints*, (1006.4357), June 2010.

[12] D. R. Dooly, S. A. Goldman, and S. D. Scott. TCP dynamic acknowledgment delay (extended abstract): Theory and practice. In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC 1998)*, pages 389–398, 1998.

[13] D. R. Dooly, S. A. Goldman, and S. D. Scott. On-line analysis of the TCP acknowledgment delay problem. *Journal of the ACM*, 48:243–273, March 2001.

[14] S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[15] D. S. Johnson, M. Minkoff, and S. Phillips. The prize collecting Steiner tree problem: Theory and practice. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000)*, pages 760–769, 2000.

[16] S. Khanna, J. S. Naor, and D. Raz. Control message aggregation in group communication protocols. In *Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP 2002)*, pages 135–146. Springer Berlin / Heidelberg, 2002.

[17] E. Korach and N. Solel. Linear time algorithm for minimum weight Steiner tree in graphs with bounded tree-width. Technical Report CS0632, Technion - Israel Institute of Technology, June 1990.

[18] P. Korteweg, A. Marchetti-Spaccamela, L. Stougie, and A. Vitaletti. Data aggregation in sensor networks: balancing communication and delay costs. In *Proceedings of the 14th international conference on Structural information and communication complexity (SIROCCO'07)*, pages 139–150, Berlin, Heidelberg, 2007. Springer-Verlag.

[19] L. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proc. 22nd International*

Conference on Distributed Computing Systems Workshops, pages 575–578. IEEE Computer Society, 2002.

[20] R. Levi, R. O. Roundy, and D. B. Shmoys. Primal-dual algorithms for deterministic inventory problems. *Mathematics of Operations Research*, 31(2):267–284, 2006.

[21] R. Levi and M. Sviridenko. Improved approximation algorithm for the one-warehouse multi-retailer problem. In *Proc. APPROX-RANDOM 2006*, pages 188–199. Springer Berlin / Heidelberg, 2006.

[22] X. Li, X.-H. Xu, F. Zou, H. Du, P. Wan, Y. Wang, and W. Wu. A PTAS for node-weighted Steiner tree in unit disk graphs. In *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications (COCOA'09)*, pages 36–48, Berlin, Heidelberg, 2009. Springer-Verlag.

[23] Y. A. Oswald, S. Schmid, and R. Wattenhofer. Tight bounds for delay-sensitive aggregation. In *Proc. 27th ACM Symposium on Principles of Distributed Computing (PODC 2008)*, pages 195–202, 2008.

[24] C. H. Papadimitriou and E. L. Servan-Schreiber. The origins of the deadline: Optimizing communication in organizations. In G. Feichtinger, editor, *Economic Complexity*, volume 14 of *International Symposia in Economic Theory and Econometrics*, pages 159–187. Emerald Group Publishing, 2004.

[25] Z. Svitkina and É. Tardos. Facility location with hierarchical facility costs. *ACM Transactions on Algorithms*, 6(2):1–22, 2010.

[26] L. Wang and T. Jiang. An approximation scheme for some Steiner tree problems in the plane. *Networks*, 28(4):187–193, 1996.