Optimal competitiveness for Symmetric Rectilinear Steiner Arborescence and related problems

Erez Kantor[1⋆] and Shay Kutten[2⋆⋆]

[1] MIT CSAIL, Cambridge, MA erezk@csail.mit.edu,
[2] Technion, Haifa 32000, Israel. kutten@ie.technion.ac.il

Abstract. We present optimal competitive algorithms for two interrelated known problems involving Steiner Arborescence. One is the continuous problem of the Symmetric Rectilinear Steiner Arborescence ($SRSA$), whose online version was studied by Berman and Coulston as a symmetric version of the known Rectilinear Steiner Arborescence (RSA) problem. A very related, but discrete problem (studied separately in the past) is the online Multimedia Content Delivery (MCD) problem on line networks, presented originally by Papadimitriou, Ramanathan, and Rangan. An efficient content delivery was modeled as a low cost Steiner arborescence in a grid of network×time they defined. We study here the version studied by Charikar, Halperin, and Motwani (who used the same problem definitions, but removed some constraints on the inputs). The bounds on the competitive ratios introduced separately in the above papers were similar for the two problems: $O(\log N)$ for the continuous problem and $O(\log n)$ for the network problem, where N was the number of terminals to serve, and n was the size of the network. The lower bounds were $\Omega(\sqrt{\log N})$ and $\Omega(\sqrt{\log n})$ correspondingly.

Berman and Coulston conjectured that both the upper bound and the lower bound could be improved. We disprove this conjecture and close these quadratic gaps for both problems. We present deterministic algorithms that are competitive optimal: $O(\sqrt{\log N})$ for $SRSA$ and $O(\min\{\sqrt{\log n}, \sqrt{\log N}\})$ for MCD, matching the lower bounds for these two online problems. We also present a $\Omega(\sqrt[3]{\log n})$ lower bound on the competitiveness of any randomized algorithm that solves the online MCD problem.

1 Introduction

We present optimal online algorithms for two known interrelated problems involving Steiner Arborescences. The continuous one is the *Symmetric Rectilinear Steiner Arborescence (SRSA)* problem [3, 5]. The online Steiner *arborescence* problems are useful in modeling the time dimension in a process. Intuitively (see, e.g. Papadimitriou at al., [11]), directed edges represent the passing of

⋆ Supported by NSF grants Nos. CCF-1217506, CCF-0939370 and CCF-AF-0937274.
⋆⋆ Supported in part by the ISF and by the Technion Gordon Center.

time. Since there is no way to go back in time in such processes, all the directed edges are directed away from the initial state of the problem, resulting in an arborescence. Additional examples given in the literature included processes in constructing a Very Large Scale Integrated electronic circuits (VLSI), optimization problems computed in iterations (where it was not feasible to return to results of earlier iterations), dynamic programming, and problems involving DNA, see, e.g. [3, 5, 8, 2].

The $SRSA$ problem: A rectilinear line segment in the plane is either horizontal or vertical. A rectilinear path contains only rectilinear line segments. This path is also *y-monotone* if during the traversal, the y coordinates of the successive points are never decreasing. The input is a set of *requests* $\mathcal{R} = \{(x_1, y_1), ..., (x_N, y_N)\}$ called Steiner terminals (or points) in the positive quadrant of the plane. A feasible solution to the problem is a set of rectilinear segments connecting all the N terminals to the origin, where the path from the origin to each terminal is a rectilinear y-monotone path. The goal is to find a feasible solution in which the sum of lengths of all the segments is the minimum possible. If we also had require the path connecting the origin to any point to be some shortest path (both x-monotone and y-monotone), then the problem would have been referred to as the *Rectilinear Steiner Arborescence* (RSA) problem [9, 14, 3, 10, 6].

Online model: In the *online* version of $SRSA$ [3], the given points are presented to the algorithm with nondecreasing y-coordinates. After receiving a new given point (terminal), the on-line $SRSA$ algorithm must extend the existing arborescence solution to incorporate the new point. There are two limitations: (1) a line, once drawn, cannot be deleted, and (2) lines can only be drawn in the region between the previous given point y-coordinates and upwards.

A very related, but discrete problem is the online *Multimedia Content Delivery* (MCD) problem on line networks, presented originally by Papadimitriou, Ramanathan, and Rangan [11]. (The formal definitions appear in Section 2). The MCD problem considered a movie residing initially at some *origin* node and a set of requests, each arriving at some node at some time. *Serving* a request at a node v at time t meant delivering a movie copy to the requesting node v from some node u which has a copy at time t; or delivering a copy to v at some time $t' < t$ from u (which has a copy at time t') and then storing the copy at v from time t' until time t.

There are two types of costs, the *delivery cost* associated with the cost of sending a movie copy over the network edges and the *storage cost* associated with the cost of storing a copies at the nodes. MCD captured the tradeoff between the storage and the delivery costs. The goal is to serve all the requests with minimal costs. An example of an algorithm would be to store, always, a movie copy at the origin and serve every request by delivering a copy from the origin at the time of the request. Such an algorithm would incur a high delivery cost. Alternatively, a copy already delivered to some nodes, could be stored there, and delivered later from there. This could reduce delivery costs, but incur storage costs. Papadimitriou et al. defined a grid of network×time (detailed in Section 2), were a request at a node u at time t was translated into a grid point (u, t). A

copy stored at a node u they modeled as an edge along the "time dimension" in the above grid (from grid point (u,t) to grid point $(u,t+1)$), while the delivery they modeled as edges along the "network dimension". A solution (an efficient content delivery plan), was modeled as a low cost Steiner arborescence leading from the origin (node 0 at time 0) to all the requests (the Steiner points). Since time is irreversible, their Steiner tree was (semi) directed away from the origin.

Papadimitriou et al. assumed some constraints on the input. Those constraints were lifted in the paper of Charikar, Halperin and Motwani [4]. The upper bound (in Charikar et al.) on the competitive ratio was $O(\log n)$ for the network problem (where n was the size of the network) and the lower bound was $\Omega(\sqrt{\log n})$. The bounds of Berman and Coulston for $SRSA$ were very similar. The upper bound was $O(\log N)$, where N was the number of terminals[3]. The lower bound was $\Omega(\sqrt{\log N})$. Clearly, these upper bounds were quadratic in the lower bounds. Berman and Coulston conjectured that both the upper bound and the lower bound could be improved.

Our results. In this paper, we disprove the above conjecture and close these quadratic gaps for both problems. We first present an $O(\sqrt{\log n})$ deterministic competitive algorithm for MCD on the line. We then translate the online algorithm to become a competitive optimal algorithm SRSA^{on} for $SRSA$. The competitive ratio is $O(\sqrt{\log N})$. Finally, we translate SRSA^{on} back to solve the MCD problem. This reverse translation improves the upper bound to $O(\min\{\sqrt{\log n}, \sqrt{\log N}\})$. That is, this final algorithm is competitive optimal for MCD even in the case that the number of requests is small. Intuitively, the "reverse translation" gets rid of the dependance on the network size, using the fact that in the definition of $SRSA$, there is no network. (This last trick may be a useful twist on the common idea of a translation between continuous and discrete problems). We also present an $\Omega(\sqrt[3]{\log n})$ lower bound on the competitiveness of any randomized algorithm that solves the online MCD problem.

Some additional related work. As pointed out in [4], MCD also motivated as a variant of a problem that is useful for data structures for the maintenance of kinematic structures, with numerous applications. Of course, Steiner trees, in general, have many applications, see e.g. [7] for a rather early survey that already included hundreds of items. $SRSA$ is a variant of the Rectilinear Steiner Arborescence (continuous) problem RSA. The offline version of RSA was studied e.g. by Rao, Sadayappan, Hwang, and Shor [14]. RSA was attributed to [10, 6] who gave two different exponential time algorithms. PTAS for RSA and $SRSA$ were presented by [9] and [5], respectively. A generalization of the logarithmic upper bound of online MCD to general networks appears in [1].

Paper structure. Section 3 contains an optimal upper bound on the competitive ratio for MCD as a function of the network size. In Section 4, the above is translated to a tight upper bound for $SRSA$. In Section 5, we use the solution of

[3] In fact, the parameter they used was p, the *normalized* size of the network. For simplicity, we present results for n, the size of the network. However, the same results for p follow easily from Sections 4 and 5.

$SRSA$ in order to improve the solution of MCD (to be optimal also as a function of the number of terminals). Finally, Section 6 includes the lower bound for randomized algorithms. Because of space considerations, most of the details in the last three sections were moved and will appear in the full version.

2 Preliminaries

The $SRSA$ problem and its online version was given in the introduction. This section contains formal definitions and notations for the network×time grid, as well as the MCD problem and its online version on that grid. Finally, it contains the offline algorithm of [4] for MCD, which we use later as a tool.

The network×time grid A *line network* $L(n) = (V_n, E_n)$ is a network whose vertex set is $V_n = \{1, ..., n\}$ and its edge set is $E_n = \{(i, i+1) \mid i = 1, ..., n-1\}$. Given a line network $L(n) = (V_n, E_n)$, construct "time-line" graph $\mathcal{L}(n) = (\mathcal{V}_n, \mathcal{E}_n)$, intuitively, by "layering" multiple replicas of $L(n)$, one per time unit, where in addition, each node in each replica is connected to the same node in the next replica . Formally, the node set $\mathcal{V}_n$ contains a *node replica* (sometimes called just a *replica*) (v, t) of every $v \in V_n$, for every time step $t \in \mathbb{N}$. That is, $\mathcal{V}_n = \{(v, t) \mid v \in V_n, t \in \mathbb{N}\}$. The set of edges $\mathcal{E}_n = \mathcal{H}_n \cup \mathcal{A}_n$ contains *horizontal edges* $\mathcal{H}_n = \{((u, t), (v, t)) \mid (u, v) \in E_n, t \in \mathbb{N}\}$, connecting network edges in every time step (round), and directed *vertical edges*, called *arcs*, $\mathcal{A}_n = \{((v, t), (v, t+1)) \mid v \in V_n, t \in \mathbb{N}\}$, connecting different copies of V_n. When it is clear from the context, we may omit n from X_n and write just X, for every $X \in \{V, E, \mathcal{V}, \mathcal{H}, \mathcal{A}\}$. Notice that $\mathcal{L}(n)$ can be viewed geometrically as a grid of n by ∞ whose grid points are the replicas. Let $d((u, s), (v, t))$ be the distance from (u, s) to (v, t). Formally, $d((u, s), (v, t)) = t - s + |v - u|$ (if $s \leq t$, otherwise, ∞).

***MCD*:** We are given a line network $L(n)$, an *origin* node $v_0 \in V$, and a set of *requests* $\mathcal{R} \subseteq \mathcal{V}$. A feasible solution is a subset of edges $\mathcal{F} \subseteq \mathcal{E}$ such that for every request $r \in \mathcal{R}$, there exists a path in $\mathcal{F}$ from the origin $(v_0, 0)$ to r. A horizontal edge $((v, t), (v+1, t)) \in \mathcal{F} \cap \mathcal{H}$ stands for sending a copy of the movie (or *copy*, for short) from node v to node $v+1$, or from node $v+1$ to node v at time t, while a vertical (directed) edge $((v, t), (v, t+1)) \in \mathcal{F} \cap \mathcal{A}$ stands for keeping the movie in v's cache at time step t for time $t+1$. For convenience, the endpoints $\mathcal{V}_\mathcal{F}$ of edges in $\mathcal{F}$ are also considered parts of the solution. For a given algorithm A, let $\mathcal{F}_A$ be the solution of A, and let $cost(A, \mathcal{R})$, (the cost of algorithm A), be $|\mathcal{F}_A|$. The goal is to find a minimum cost feasible solution. In our analysis, OPT is the set of edges in some optimal solution whose cost is $|\text{OPT}|$.

Online model In the online versions of the problem, the algorithm receives as input a sequence of events. One type of events is a request in the (ordered) set $\mathcal{R}$ of requests $\mathcal{R} = \{r_1, r_2, ..., r_N\}$ (like in $SRSA$). A second type of events is a time event (this event does not exists in $SRSA$), where we assume a clock that tells the algorithm that no additional requests for time t are about to arrive (or that there are no requests for some time t at all). The algorithm then still has the opportunity to complete its calculation for time t (e.g., add arcs from some replica (v, t) to $(v, t+1)$). Then time $t+1$ arrives.

When handling an event ev, the algorithm only knows the following: (a) all the previous requests $r_1, ..., r_i$; (b) time t; and (c) the solution arborescence $\mathcal{F}_{ev}$ it constructed so far (originally containing only the origin). In each event, the algorithm may need to make decisions of two types, before seeing future events:

(1.MCD) If the event is the arrival of a request $r_i = (v_i, t_i)$, then from which *current* (time t_i) cache (a point already in the solution arborescence $\mathcal{F}_{ev}$ when r_i arrives) to serve r_i by adding horizontal edges to $\mathcal{F}_{ev}$.

(2.MCD) If this is the time event for time t, then at which nodes to store a copy for time $t+1$, for future use: select some replica (or replicas) (v,t) already in the solution $\mathcal{F}_{ev}$ and add to $\mathcal{F}_{ev}$ an edge directed from (v,t) to $(v,t+1)$.

Note that, at time t, the online algorithm cannot add nor delete any edge with an endpoint that corresponds to previous times. Similarly to e.g. [1, 11, 13, 12, 4], we assume that at least one copy must remain in the network at all times[4].

A tool: the offline algorithm TRIANGLE **of Charikar et al.** Consider a request set $\mathcal{R} = \{r_0 = (v_0, 0), r_1 = (v_1, t_1), ..., r_N = (v_N, t_N)\}$ such that $0 \leq t_1 \leq t_2 \leq ... \leq t_N$. When Algorithm TRIANGLE starts, the solution includes just $r_0 = (v_0, 0)$ (intuitively, a "pseudo request"). Then, TRIANGLE handles, first, request r_1, then request r_2, etc... In handling a request r_i, the algorithm may add some (possibly "past") edges to the solution. (It never deletes any edge from the solution.) After handling r_i, the solution is an arborescence rooted at r_0 that spans the request replicas $r_1, ..., r_i$. For each such request $r_i \in \mathcal{R}$, TRIANGLE performs the following.

(T1) Choose a replica $q_i^{\mathrm{T}} = (u_i^{\mathrm{T}}, s_i^{\mathrm{T}})$ s.t. q_i^{T} is already in the solution and the distance from q_i^{T} to r_i is minimum (over the replicas already in the solution). Call q_i^{T} the *serving replica* of r_i.

(T2) Define the *radius* ρ_i^{T} of r_i as $\rho_i^{\mathrm{T}} = d(q_i^{\mathrm{T}}, r_i)$. Also define the *base*[5] $\textsc{Base}(i)$ of r_i as the set of replicas at time t_i of distance at most ρ_i^{T} from r_i. That is, $\textsc{Base}(i) = \{q = (v, t_i) \in \mathcal{V} \mid d(r_i, q) \leq \rho_i^{\mathrm{T}}\}$. Similarly, the *edge base* of r_i is $\textsc{Base}_{\mathcal{H}}(i) = \{(r, q) \in \mathcal{H} \mid r, q \in \textsc{Base}(i)\}$.

(T3) Deliver a copy to each replica in $\textsc{Base}(i)$. That is, node u_i^{T} stores a copy from time s_i^{T} to time t_i. More formally, add the arcs of $\mathcal{P}_{\mathcal{A}}[(u_i^{\mathrm{T}}, s_i^{\mathrm{T}}), (u_i^{\mathrm{T}}, t_i)] = \{((u_i^{\mathrm{T}}, z), (u_i^{\mathrm{T}}, z+1)) \mid s_i^{\mathrm{T}} \leq z < t_i\}$ to the solution.

(T4) Deliver a copy to all replicas in $\textsc{Base}(i)$. That is, add all the edges of $\textsc{Base}_{\mathcal{H}}(i)$ to the solution, except the ones that close circle[6] (if such exists).

It is easy to verify [4] that the cost of TRIANGLE for serving r_i is at most $3\rho_i^{\mathrm{T}}$. Denote by $\mathcal{F}^{\mathrm{T}} = \mathcal{H}^{\mathrm{T}} \cup \mathcal{A}^{\mathrm{T}}$ the feasible solution of TRIANGLE, where $\mathcal{H}^{\mathrm{T}} \subseteq \cup_{i=1}^{N} \textsc{Base}_{\mathcal{H}}(i)$ and $\mathcal{A}^{\mathrm{T}} = \cup_{i=1}^{N} \mathcal{P}_{\mathcal{A}}[(u_i^{\mathrm{T}}, s_i^{\mathrm{T}}), (u_i^{\mathrm{T}}, t_i)]$. Note that $\mathcal{F}^{\mathrm{T}}$ is an arborescence rooted at $(v_0, 0)$ spanning the base replicas of $\textsc{Base} = \cup_{i=1}^{N} \textsc{Base}(i)$. Rewording the theorem of [4], somewhat,

[4] Alternatively, the system (not the algorithm) can have the option to delete the movie altogether, this decision must then be made known to the algorithm. At least one of these natural assumptions is also necessary for having a competitive algorithm.

[5] The word "base" comes from the notation used in [4] for Algorithm TRIANGLE. There, $\textsc{Base}(i)$ is a base of the triangle.

[6] For convenience of the analysis we want the solution to be a tree.

Theorem 21 *[4] $\mathcal{F}^{\mathrm{T}}$ is a 3-approximate solution. Also, $\sum_{i=1}^{N} \rho_i^{\mathrm{T}} \leq |\mathrm{OPT}|$.*

3 Optimal online algorithm for MCD

Algorithm LINE$^{\mathbf{on}}$. Like Algorithm TRIANGLE, Algorithm LINE$^{\mathrm{on}}$ handles requests one by one, according to the order of arrival. However, in step (T3), TRIANGLE may perform an operation that no online algorithm can perform (if $s_i^{\mathrm{T}} < t_i$). Serving a request r_i must be performed from some replica $q_i^{\mathrm{on}} = (u_i^{\mathrm{on}}, t_i) \in \mathcal{V}[t_i]$ that holds a copy at time t_i in the execution of the online algorithm on $\mathcal{R}$. Thus (in addition to selecting from which nodes to deliver copies), algorithm LINE$^{\mathrm{on}}$ at time $t_i - 1$ had to also select the nodes that store copies for the consecutive time t_i (so that q_i^{on} mentioned above would be one of them). Let us start with some definitions.

General definitions and notations. Consider an interval $J = \{v, v+1, ..., v+\rho\} \subseteq V$ and two integers $s, t \in \mathbb{N}$, s.t. $s \leq t$. Let $J[s,t]$ be the *"rectangle subgraph"* of $\mathcal{L}(n)$ corresponding to vertex set J and time interval $[s,t]$. This rectangle consists of the replicas and edges of the nodes of J corresponding to time interval $[s,t]$. For a given subsets $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{H}' \subseteq \mathcal{H}$ and $\mathcal{A}' \subseteq \mathcal{A}$, denote by (1) $\mathcal{V}'[s,t]$ replicas of $\mathcal{V}'$ corresponding to times $s, ..., t$. Define similarly (2) $\mathcal{H}'[s,t]$ for horizontal edges of $\mathcal{H}'$; and (3) $\mathcal{A}'[s,t]$ arcs of $\mathcal{A}'$. (When $s = t$, we may write $\mathcal{X}[t] = \mathcal{X}[s,t]$, for $\mathcal{X} \in \{J, \mathcal{V}', \mathcal{H}'\}$.) Consider also two nodes $v, u \in V$. Let $\mathcal{P}_{\mathcal{H}}[(v,t),(u,t)] = \mathcal{P}_{\mathcal{H}}[(u,t),(v,t)]$ be the set of horizontal edges of the shortest path from (v,t) to (u,t).

Partitions of $[1,n]$ into intervals. Define $m = n/\Delta$ for some positive integer Δ to be chosen later. For convenience, we assume that $m = n/\Delta$ is a power of 2. (It is trivial to generalize it). Define $\log m + 1$ *levels* of partitions of the interval $[1,n]$. In level l, partition $[1,n]$ into $m/2^l = n/\Delta 2^l$ intervals, I_1^l, I_2^l,...,$I_{m/2^l}^l$, each of size $\Delta 2^l$. $I_j^l = \{\Delta(j-1) \cdot 2^l + k \mid k = 1, ..., \Delta 2^l\}$, for every $1 \leq j \leq m/2^l$ and every $0 \leq l \leq \log m$. Let $\mathcal{I}$ be the set of all such intervals. Let $\ell(I)$ be the *level* of an interval $I \in \mathcal{I}$, i.e., $\ell(I_j^l) = l$. We say that I is a level $\ell(I)$ interval. Denote by $I^l(v)$ (for every node $v \in V$ and every level $l = 0, ..., \log m$) the interval in level l that contains v. That is, $I^l(v) = I_k^l$, where $k = \left\lfloor \frac{v}{\Delta 2^l} \right\rfloor + 1$.

For a given interval $I_j^l \in \mathcal{I}$, denote by $N^R(I_j^l)$, for $1 \leq j < m/2^l$ (respectively, $N^L(I_j^l)$, for $1 < j \leq m/2^l$) the *neighbor* interval of level l that is on the right (resp., left) of I_j^l. That is, $N^L(I_j^l) = I_{j-1}^l$ and $N^R(I_j^l) = I_{j+1}^l$. Define that $N^L(I_1^i) = \emptyset$ and $N^R(I_{m/2^l}^i) = \emptyset$. Let $N(I) = N^L(I) \cup I \cup N^R(I)$. We say that $N(I)$ is the *neighborhood* of I.

Active intervals. An interval $I \in \mathcal{I}$ is called *active* at time t, if $\mathrm{BASE} \cap I[t - 2^{\ell(I)}, t] \neq \emptyset$. Intuitively, TRIANGLE kept a movie copy in, at least, one of the nodes of I, at least once, and "not too long" before time t. We say that I *stays-active*, intuitively, if I is **not** "just about to stop being active", that is, if $\mathrm{BASE} \cap I[t - 2^{\ell(I)} + 1, t] \neq \emptyset$.

Denote by $\mathcal{C}_{t+1}$ the set of replicas corresponding to the nodes that store copies from time t to time $t+1$ in a LINE$^{\mathrm{on}}$ execution. Also, $\mathcal{C}_0 = \{r_0 = (v_0, 0)\}$

(we choose to store a copy in v_0 always). To help us later in the analysis, we also added an auxiliary set COMMIT $\subseteq \{\langle I,t\rangle \mid I \in \mathcal{I}, t \in \mathbb{N}\}$. Initially, COMMIT $\leftarrow \emptyset$. For each time $t = 0, 1, 2, ...$, consider first the case that there exists at least one request corresponding to time t, i.e., $\mathcal{R}[t] = \{r_j, ..., r_k\} \neq \emptyset$. Then, for each request $r_i \in \mathcal{R}[t]$, LINE$^{\text{on}}$ simulates TRIANGLE to find the radius ρ_i^{T} and the set of base replicas BASE(i) of r_i. Next, LINE$^{\text{on}}$ delivers a copy to every such base replica $r \in$ BASE(i) (this is called the *"delivery phase"*). That is, for each $i = j, ..., k$ do:

(D1) choose a closest (to r_i) replica $q_i^{\text{on}} = (u_i^{\text{on}}, t)$ of time t already in the solution;
(D2) add the path $\mathcal{H}^{\text{on}}(i) = \mathcal{P}_{\mathcal{H}}[q_i^{\text{on}}, r_i] \cup \text{BASE}_{\mathcal{H}}(i)$ to the solution.

Let $\mathcal{V}^{\text{on}}(i) = \{r \mid (r,q) \in \mathcal{H}^{\text{on}}(i)\}$. (Note that r_j is served from $\mathcal{C}_t$, after that, r_{j+1} is served from $\mathcal{C}_t \cup \mathcal{V}^{\text{on}}(j)$, etc.) Clearly, the delivery phase of time t ensures that (at least) the nodes of $\mathcal{C}_t \cup$ BASE$[t]$ have copies at the end of that phase. It is left to decide which of the above copies to store for time $t+1$. That is (the *"storage phase"*), LINE$^{\text{on}}$ chooses the set $\mathcal{C}_{t+1} \subseteq \mathcal{C}_t \cup$ BASE$[t]$. Initially, $C_{t+1} \leftarrow \{(v_0, t+1)\}$ (as we choose to store a copy at v_0). Then, for each level $l = 0, ..., \log m$, in an *increasing* order, select as follows.

(S1) While there exists a level l interval $I \in \mathcal{I}$ that is (i) stays-active at t; but (ii) no replica has been selected in I's neighborhood (i.e., $\mathcal{C}_{t+1} \cap N(I)[t+1] = \emptyset$), then perform steps (S1.1-S1.3) below.
(S1.1) Add the tuple $\langle I, t\rangle$ to the set COMMIT (we say that I *commits* at time t).
(S1.2) Select some replica $(v,t) \in$ BASE$[t] \cup \mathcal{C}_t$ such that $v \in N(I)$ (by Observation 1 below, such a replica does exist).
(S1.3) Add $(v, t+1)$ to $\mathcal{C}_{t+1}$ and add the arc $((v,t),(v,t+1))$ to the solution.

The pseudo code of LINE$^{\text{on}}$ and an example for an execution of LINE$^{\text{on}}$ are given in the full version. The solution constructed by LINE$^{\text{on}}$ is denoted $\mathcal{F}^{\text{on}} = \mathcal{H}^{\text{on}} \cup \mathcal{A}^{\text{on}}$, where $\mathcal{H}^{\text{on}} = \cup_{i=1}^{N} \mathcal{H}^{\text{on}}(i)$ represents the horizontal edges added in the delivery phases and $\mathcal{A}^{\text{on}} = \{((v,t),(v,t+1)) \mid (v,t+1) \in \mathcal{C}_{t+1} \text{ and } t = 0, ..., t_N\}$ represents the arcs added in the storage phase. Before the main analysis, we make some easy-to-prove but crucial observations. For completeness, their proofs appear in full version (however, they are pretty clear from step S1). Recall that the notation of active (including stays-active) refer to the fact that the nodes of some base replicas belong to some interval I in the ("recent") past. Observations 1 and 2 state, intuitively, that LINE$^{\text{on}}$ leaves a copy in the *neighborhood* $N(I)$ of I as long as I is active.

Observation 1 ("WELL DEFINED"). *If an interval $I \in \mathcal{I}$ is stays-active at time t, then there exists a replica $(v,t) \in \mathcal{C}_t \cup$ BASE$[t]$ such that $v \in N(I)$.*

Observation 2 ("AN ACTIVE INTERVAL HAS A NEARBY COPY"). *If an interval I is active at time t, then, either (i) there is some base replica in I's neighborhood at t (BASE$\cap N(I)[t] \neq \emptyset$), or (ii) at least one of the nodes of $N(I)$ stores a copy for time t ($N(I)[t] \cap \mathcal{C}_t \neq \emptyset$).*

Observation 3 ("BOUND FROM ABOVE ON $|\mathcal{A}^{\text{on}}|$"). $|\mathcal{A}^{\text{on}}| \leq |\text{COMMIT}| + t_N$.

3.1 Analysis of LINE$^{\text{on}}$

We, actually, prove that $\frac{cost(\text{LINE}^{\text{on}},\mathcal{R})}{cost(\text{TRIANGLE},\mathcal{R})} = O(\sqrt{\log n})$. This implies the desired competitive ratio of $O(\sqrt{\log n})$ by Theorem 21. We first show, that the number of horizontal edges in $\mathcal{H}^{\text{on}}$ (*"delivery cost"*) is $O\left(\Delta \cdot cost(\text{TRIANGLE},\mathcal{R})\right)$. Then, we show, that the the number of arcs in $\mathcal{A}^{\text{on}}$ (*"storage cost"*) is $O(\frac{\log n}{\Delta} \cdot cost(\text{TRIANGLE},\mathcal{R}))$. Optimizing Δ, we get a competitiveness of $O(\sqrt{\log n})$.

Delivery cost analysis. For each request $r_i \in \mathcal{R}$, the delivery phase (step (D2)) adds $\mathcal{H}^{\text{on}}(i) = \mathcal{P}_{\mathcal{H}}[q_i^{\text{on}}, r_i] \cup \text{BASE}_{\mathcal{H}}(i)$ to the solution. Define the *online* radius of r_i as $\rho_i^{\text{on}} = d(q_i^{\text{on}}, r_i)$. Since $|\text{BASE}_{\mathcal{H}}(i)| \leq 2\rho_i^{\text{T}}$, it follows that,

$$|\mathcal{H}^{\text{on}}| \leq \sum_{i=1}^{N} \left(\rho_i^{\text{on}} + 2\rho_i^{\text{T}}\right). \tag{1}$$

It remains to bound ρ_i^{on} as a function of ρ_i^{T} from above. Intuitively, ρ_i^{T} includes the distance from some base replica $q_i = (u_i, s_i) \in \text{BASE}$ to $r_i = (v_i, t_i)$. That is, ρ_i^{T} includes the distance from v_i to u_i and the time difference between s_i and t_i. Restating Observation 2 somewhat differently (Claim 4 below), we can use the distance $|v_i - u_i| \leq \rho_i^{\text{T}}$ and the time difference $t_i - s_i \leq \rho_i^{\text{T}}$ for bounding ρ_i^{on}. That is, we show that LINE$^{\text{on}}$ has a copy at time t_i (of r_i) at a distance at most $4\Delta\rho_i^{\text{T}}$ from u_i (of q_i). Since, $|v_i, u_i| \leq \rho_i^{\text{T}}$, LINE$^{\text{on}}$ has a copy at distance at most $(4\Delta + 1)\rho_i^{\text{T}}$ from v_i (of r_i). Throughout, since lack of space some of the proofs are omitted.

Lemma 4 *Consider some base replica $(v,t) \in \text{BASE}$ and some $\rho > 0$, such that, $t+\rho \leq t_N$. Then, there exists a replica $(w, t+\rho) \in \mathcal{C}_{t+\rho}$ such that $|v-w| \leq 4\Delta\rho$.*

Lemma 5 $\rho_i^{\text{on}} \leq (4\Delta + 1) \cdot \rho_i^{\text{T}}$.

The following corollary follows from the above lemma, Ineq. (1) and Theorem 21.

Corollary 1. $|\mathcal{H}^{\text{on}}| \leq (4\Delta + 3) \cdot |\text{OPT}|$.

Storage cost analysis. By Observation 3, it remains to bound the size of |COMMIT| from above. Let $commit(I,t) = 1$ if $\langle I, t\rangle \in \text{COMMIT}$ (otherwise 0). Hence, $|\text{COMMIT}| = \sum_{I\in\mathcal{I}} \sum_{t=0}^{\infty} commit(I,t)$. We begin by bounding the number of commitments in LINE$^{\text{on}}$ made by level $l = 0$ intervals.

Observation 6 $\sum_{I\in\{J\in\mathcal{I}|\ell(J)=0\}} commit(I,t) \leq \left|\text{BASE}\right|$.

The following is our main lemma;

Lemma 7 $|\text{COMMIT}| \leq 3\left|\mathcal{A}^{\text{T}}\right| + \frac{6\log n}{\Delta}\left|\mathcal{H}^{\text{T}}\right| + |\text{BASE}|$.

Proof sketch. The |BASE| term in the statement of the lemma follows from Observation 6 for level $l = 0$ intervals. The rest of the proof deals with commitments in intervals $I \in \mathcal{I}$ whose level $\ell(I) > 0$. We now group the commitments of each such an interval into *"bins"*. Later, we shall "charge" the commitments in each bin on certain costs of the offline algorithm TRIANGLE.

Consider an input $\mathcal{R}$ and some interval $I \in \mathcal{I}$ of level $\ell(I) > 0$. We say that I is a *committed-interval* if I commits at least once in the execution of LINE^{on} on $\mathcal{R}$. For each committed-interval I (of level $\ell(I) > 0$), we define (almost) non-overlapping *"sessions"* (one session may end at the same time the next session starts; hence, two consecutive sessions may overlap on their boundaries). The first session of I does *not* contain any commitments (and is termed an *uncommitted-session*); it begins at time 0 and ends at the first time that I contains some base replica. Every other session (of I) contains at least one commitment (and is termed a *committed-session*).

Each commitment (in LINE^{on}) of I belongs to some committed session. Given a commitment $\langle I, t\rangle \in \text{COMMIT}$ that I makes at time t, let us identify $\langle I, t\rangle$'s session. Let $t^- < t$ be the last time (before t) there was a base replica in I. Similarly, let $t^+ > t$ be the next time (after t) there will be a base replica in I (if such a time does exist; otherwise, $t^+ = \infty$). The session of commitment $\langle I, t\rangle$ starts at t^- and ends at t^+. Similarly, when talking about the i's session of interval I, we say that the session starts at $t_i^-(I)$ and ends at $t_i^+(I)$. When I is clear from the context, we may omit (I) and write t_i^-, t_i^+. A bin is a couple (I, i) of a commitment-interval and the ith commitment-session of I. Clearly, we assigned all the commitments (of level $l > 0$ intervals) into bins.

Observation 8 *The bins do not overlap (except, perhaps, on their boundaries).*

Let us now point at costs of algorithm TRIANGLE on which we shall "charge" the set of commitments $\text{COMMIT}(I, i)$ in bin (I, i). We now consider only a bin (I, i) whose committed session is not the last. Note that the bin corresponds to a rectangle of $|I|$ by $t_i^+ - t_i^-$ replicas. Expand the bin by $|I|$ replicas left and $|I|$ replicas right, if such exist (to I's neighborhood $N(I)$). This yields the *payer* of bin (I, i); that is the payer is a rectangle subgraph of $|N(I)|$ by $t_i^+ - t_i^-$ replicas. We point at specific costs TRIANGLE had in this payer.

Recall that every session of I, except may the last, must ends with a base replica in I. Let $(v, t_i^+) \in \text{BASE} \cap I[t_i^+]$ be some base replica in I at the ending time of the session. The solution of TRIANGLE must contain a route (TRIANGLE route) that starts at the root and reaches (v, t_i^+) by the definition of a base replica. For the charging, we use *some* (detailed below) of the edges in the intersection of the TRIANGLE route and the payer rectangle.

The easiest case (**EB**, for Entrance from Below) is that the TRIANGLE route enters the payer at the payer's bottom (t_i^-) and stays in the payer until t_i^+. Then, each time $(t_i^- < t < t_i^+)$ there is a commitment in the bin, there is also an arc a_t in the TRIANGLE route (from time t to time $t+1$). We charge that commitment on that arc a_t. Intuitively, the same arc a_t may be charged also for one bin on the left of (I, i) and one bin on its right, since the payer rectangles are 3 times wider than the bins. Note that arc a_t may also belong to additional $O(\log n)$ payers (of

bins of intervals that contain I or are contained in I). The crucial point is that a_t is *not* charged for those additional bins. That is, we claim that there are no commitments for those other bins. Intuitively, LINE$^{\text{on}}$ was designed such that if I commits at time t, LINE$^{\text{on}}$ also stores a copy in I's neighborhood for time $t+1$. Hence, an interval J whose neighborhood contains the neighborhood of I, does not need to commit (and the test fails in (S1) in LINE$^{\text{on}}$). Thus, an arc of the TRIANGLE route is charged only by 3 commitments at most.

In the remaining case (**SE**, for Side Entrance), the TRIANGLE route enters the payer from either the left or the right side of the payer. (That is, TRIANGLE delivers a copy from some other node u outside I's neighborhood, rather than stores copies at I's neighborhood from some earlier time. Therefore, the route must "cross" either the left neighbor interval of I or the right neighbor interval in that payer. Thus, there exists at least $|I| = \Delta 2^{\ell(I)}$ horizontal edges in the intersection between the payer ($payer(I,i)$), of (I,i) and the TRIANGLE route. On the other hand, the number of commitments in bin (I,i) is $2^{\ell(I)}$ at most. (To commit, an interval must be active; to be active, it needs a base replica in the last $2^{\ell(i)}$ times; a new base replica would end the session.) That is, we charged the payer Δ times more horizontal edges than there are commitments in the bin. On the other hand, each horizontal edge participates in $O(\log n)$ payers (payers of 3 intervals at most in each level; and payers of 2 bins of each interval at most, since two consecutive sessions may intersect only at their boundaries). This leads to the term $\frac{6\log n}{\Delta}$ before the $|\mathcal{H}^{\text{T}}|$ in the statement of the lemma.

For each interval I, it is left to account for commitments in I's last session. That is, we now handle the bin (I,i') where I has i' commitment-sessions. This session may not end with a base replica in I, so we cannot apply the argument above that TRIANGLE must have a route reaching a replica in I at $t^+_{i'}$. On the other hand, the first session of I (the uncommitted-session) does end with a base replica in I, but has no commitments. Intuitively, we use the payer of the first session of I to pay for the commitments of the last session of I. Specifically, in the first session, the TRIANGLE route must enter the neighborhood of I from the side; (Note that the TRIANGLE route still starts outside I; this because the origin v_0 who holds a copy, is not in I's neighborhood; otherwise, I would not have been a committed interval.) Hence, we apply the argument of case SE above. *(End of Proof sketch.)* ∎

We now optimize a tradeoff between the storage coast and the delivery cost of LINE$^{\text{on}}$. On the one hand, Lemma 7 shows that a large Δ reduces the number of commitments. By Observation 3, this means a large Δ reduces the storage cost of LINE$^{\text{on}}$. On the other hand, corollary 1 shows that a *small* Δ reduces the delivery cost. To balance this tradeoff, we need to "manipulate" Lemma 7 somewhat, since it uses variables that are different from those used in corollary 1. We use the following observation (1) $t_N \leq |\text{OPT}| \leq cost(\text{TRIANGLE}, \mathcal{R})$; (2) $|\mathcal{A}^{\text{T}}| + |\mathcal{H}^{\text{T}}| = cost(\text{TRIANGLE}, \mathcal{R})$; and (3) $|\text{BASE}| \leq cost(\text{TRIANGLE}, \mathcal{R})$. Substituting the above (1)–(3) in Observation 3 and Lemma 7,

$$|\mathcal{A}^{\text{on}}| \leq \left(5 + \frac{3\log n}{\Delta}\right) \cdot cost(\text{TRIANGLE}, \mathcal{R}). \tag{2}$$

To optimize the tradeoff, fix $\Delta = \sqrt{10 \log n}$. Corollary 1, and inequality (2) imply that $cost(\text{LINE}^{\text{on}}, \mathcal{R}) = |\mathcal{A}^{\text{on}}| + |\mathcal{H}^{\text{on}}| \leq (8 + \sqrt{10 \log n}) \cdot cost(\text{TRIANGLE}, \mathcal{R})$. Thus, by Theorem 21, the following holds.

Theorem 31 LINE$^{\text{on}}$ *is* $O(\sqrt{\log n})$*-competitive for MCD on the line network.*

4 Optimal online algorithm for *SRSA*

Note that our solution for MCD (Section 3) does not yet solve $SRSA$. In MCD, the X coordinate of every request (in the set $\mathcal{R}$) is taken from a known set of size n (the network nodes $\{1, 2, ..., n\}$). On the other hand, in $SRSA$, the X coordinate of a *point* is arbitrary. Let us now transform, in three concenptual stages LINE$^{\text{on}}$ into an optimal algorithm for the online problem of $SRSA$:

1. Given an instance of $SRSA$, assume temporarily (and remove the assumption later) that the number N of points is known, as well as M, the maximum X coordinate any request may have. Then, simulate a network where $n \geq N$ and $n = O(\sqrt{\log N})$, and the n nodes are spaced evenly on the interval between 0 and M. Transform each $SRSA$ request to the nearest grid point. Solve the resulting MCD problem.
2. Translate these results to results of the original $SRSA$ instance.
3. Get rid of the assumptions.

% The first stage is easy. It turns out that "getting rid of the assumptions" is also relatively easy. To simulate the assumption that M is known, guess that M is some M_j. Whenever a guess fails, (a request $r_i = (x_i, t_i)$ arrives, where $x_i > M_j$), continue with an increased guess M_{j+1}. A similar trick is used for guessing N. In implementing this idea, our algorithm turned out paying a cost of ΣM_j (M_j for a failed guess), while an algorithm that knew M could pay M only once. IF M_{j+1} is "sufficiently" larger than M_j, then $\Sigma M_j = O(M)$. The "sufficiently larger" part turned out somewhat trickier for guessing N than for guessing M.

The second stage above (translate the results) proved to be more difficult, even in the case that N and M are known (and even equal). Intuitively, following the first stage, each request $r_i = (x_i, t_i)$ is in some grid square, where the corners of the square are points of the simulated MCD problem. If we normalize M to be N, then the left bottom left corner of that square is $(\lfloor x_i \rfloor, \lfloor t_i \rfloor))$. Had we wanted an *offline* algorithm, we could have solved an instance of MCD, where the points are $(\lfloor x_1 \rfloor, \lfloor t_1 \rfloor), (\lfloor x_2 \rfloor, \lfloor t_2 \rfloor), (\lfloor x_3 \rfloor, \lfloor t_3 \rfloor), ...$ Then, translating the results of MCD would have meant just augmenting with segments connecting each $(\lfloor x_i \rfloor, \lfloor t_i \rfloor)$ to (x_i, t_i). Unfortunately, this is not possible in an *online* algorithm, since (x_i, t_i) is not yet known at $(\lfloor t_i \rfloor)$. Similarly, we cannot use the upper left corner of the square (for example) that way, since at time $\lceil t_i \rceil$, the algorithm may no longer be allowed to add segments reaching the earlier time t_i. Because of the lack of space, we moved the rest of this proof.

Theorem 41 *Algorithm* SRSA$^{\text{on}}$ *is optimal and is* $O(\sqrt{\log N})$*-competitive.*

5 Optimizing *MCD* for a small number of requests

Algorithm LINE$^{\text{on}}$ was optimal only as the function of the network size (Theorem 31). Recall that our solution for $SRSA$ was optimal as a function of the number of requests. We transform that algorithm back to solve MCD, and obtain the promised competitiveness, $O(\min\{\sqrt{\log n}, \sqrt{\log N}\})$.

6 Randomized Lower Bound for Line Networks

Our lower bound on the competitive ratio of randomized algorithms then follows from Yaos min-max principle [15] appears in the full version.

Theorem 61 *The competitive ratio of any randomized online algorithm for MCD on line networks $\Omega(\sqrt[3]{\log n})$.*

Acknowledgment We would like to thank to Reuven Bar-Yehuda and Dror Rawitz for insights and helpful dissections.

References

1. R. Bar-Yehuda, E. Kantor, S. Kutten, and D. Rawitz. Growing half-balls: Minimizing storage and communication costs in cdns. In *ICALP*, pages 416–427, 2012.
2. W. Bein, M. Golin, L. Larmore, and Y. Zhang. The knuth-yao quadrangle-inequality speedup is a consequence of total monotonicity. *TOPLAS*, 6(1), 2009.
3. P. Berman and C. Coulston. On-line algorrithms for steiner tree problems. In *STOC'97*, pages 344–353, 1997.
4. M. Charikar, D. Halperin, and R. Motwani. The dynamic servers problem. In *SODA'98*, pages 410–419, 1998.
5. X. Cheng, B. Dasgupta, and B. Lu. Polynomial time approximation scheme for symmetric rectilinear steiner arborescence problem. *J. Global Optim.*, 21(4):385–396, 2001.
6. R. R. Ladeira de Matos. A rectilinear arborescence problem. *Dissertation, University of Alabama*, 1979.
7. D.S. Richards F.K. Hwang. Steiner tree problems. *Networks*, 22(1):55–897, 1992.
8. A. Kahng and G. Robins. On optimal interconnects for VLSI. *Kluwer*, 1995.
9. B. Lu and L. Ruan. Polynomial time approximation scheme for rectilinear steiner arborescence problem. *Combinatorial Optimization*, 4(3):357–363, 2000.
10. L. Nastansky, S. M. Selkow, and N. F. Stewart. Cost minimum trees in directed acyclic graphs. *Z. Oper. Res.*, 18:59–67, 1974.
11. C.H. Papadimitriou, S. Ramanathan, and P.V. Rangan. Information caching for delivery of personalized video programs for home entertainment channels. In *IEEE International Conf. on Multimedia Computing and Systems*, pages 214–223, 1994.
12. C.H. Papadimitriou, S. Ramanathan, and P.V. Rangan. Optimal information delivery. In *6th ISAAC*, pages 181–187, 1995.
13. C.H. Papadimitriou, S. Ramanathan, P.V. Rangan, and S. Sampathkumar. Multimedia information caching for personalized video-on demand. *Computer Communications*, 18(3):204–216, 1995.
14. S. Rao, P. Sadayappan, F. Hwang, and P. Shor. The rectilinear steiner arborescence problem. *Algorithmica*, pages 277–288, 1992.
15. Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *FOCS'77*, pages 222–227, 1977.