Resource Minimization for Fire Containment

Parinya Chalermsook * Julia Chuzhoy †

Abstract

We consider the following model for fire containment. We are given an undirected graph $G = (V, E)$ with a source vertex s where the fire starts. At each time step, the firefighters can save up to k vertices of the graph, while the fire spreads from burning vertices to all their neighbors that have not been saved so far. Our goal is to choose the vertices to be saved at each time step so as to contain the fire. This is a simple mathematical model abstracting the dynamic nature of fire containment and other natural processes, such as, for example, the spread of a perfectly contagious disease and its containment via vaccination.

We focus on the Resource Minimization Fire Containment (RMFC) problem, where we are additionally given a subset $T \subseteq V$ of vertices called terminals that need to be protected from fire. The objective is to minimize k - the maximum number of vertices to be saved at any time step, so that the fire does not spread to the vertices of T. The problem is hard to approximate up to any factor better than 2 even on trees. We show an $O(\log^* n)$-approximation LP-rounding algorithm for RMFC on trees. We also show that an even stronger LP relaxation has an integrality gap of $\Omega(\log^* n)$ on trees. Finally, we consider RMFC on directed layered graphs, and show an $O(\log n)$-approximation LP-rounding algorithm, matching the integrality gap of the LP relaxation.

1 Introduction

We study the following model for fire containment. The input is an undirected graph $G = (V, E)$ with a source vertex s, where the fire starts. At each time step we can choose up to k vertices to be saved by the firefighters, while the fire spreads to every vertex that has not been saved so far and has at least one burning neighbor. Once a vertex is saved or burns, it remains in this state permanently. The process stops when the fire cannot spread to any new vertices. This is a simple mathematical model for containing a natural process with a simple spread mechanism. It was first suggested by Hartnell [8] in the context of firefighting, and can also be used in a variety of similar scenarios, such as, for example, in containing an outbreak of a perfectly contagious disease via vaccination, when only a small number of individuals can be vaccinated at each time step [10].

We consider the resource minimization version of the problem, called Resource Minimization Fire Containment (RMFC). In this problem, we are also given a subset $T \subseteq V$ of vertices called terminals. The goal is to minimize k, the maximum number of vertices to be saved at any time step, so that the fire does not spread to the vertices of T. We sometimes refer to k as the number of firefighters the solution uses. Another natural version of the problem, that has been studied in the literature, is the Firefighters problem, where the bound k is fixed, and the goal is to maximize the total number of vertices to which the fire does not spread.

King and MacGillivray [13] showed that RMFC is NP-hard even on full trees of degree three, and their result implies that the problem is hard to approximate up to any factor better than 2. This is the only currently known lower bound on the approximability of RMFC, even for general graphs. On the algorithmic side, the standard randomized rounding of a natural LP-relaxation for RMFC on trees gives an $O(\log n)$-approximation. King and MacGillivray [13] also show that the problem is efficiently solvable on graphs of maximum degree 3 if the fire starts at a degree-2 vertex.

Better approximation algorithms are known for the Firefighters problem on trees. Hartnell and Li [11] showed that a simple greedy algorithm gives a factor 2-approximation. This was improved to factor $e/(e-1)$ by to Cai, Verbin and Yang [3], who also showed an exact algorithm with running time $2^{O(\sqrt{n}\log n)}$. Finbow et. al. [6] showed that the Firefighters problem is NP-hard even on trees of degree at most three, but is efficiently solvable if the fire starts at a degree-2 vertex.

Our main result is an $O(\log^* n)$-approximation algorithm for RMFC on trees. The algorithm is based on the rounding of a natural linear programming relaxation for the problem. We also show that an even stronger LP-relaxation of RMFC on trees has an $\Omega(\log^* n)$ integrality gap. Next we consider RMFC on directed layered graphs and show an $O(\log n)$-approximation LP-rounding algorithm, matching the lower bound on the integrality gap of the LP that was shown by [12].

We note that the approximation factor of $O(\log^* n)$

*Department of Computer Science, University of Chicago, Chicago, IL 60637. Email: parinya@uchicago.edu

†Toyota Technological Institute, Chicago, IL 60637. Email: cjulia@tti-c.org. Supported in part by NSF CAREER grant CCF-0844872

is rather unusual, and the only known natural problem with $\Theta(\log^* n)$ approximability threshold is Asymmetric k-Center [19, 2, 4]. This problem appears to be somewhat similar in nature to RMFC on trees: the $O(\log^* n)$-factor approximation algorithm for Asymmetric k-Center starts by transforming the problem into a covering problem on a layered graph, where every pair of consecutive layers is viewed as an instance of the Set Cover problem. While RMFC on trees can also be seen as a covering problem on layered graphs, the $O(\log^* n)$ approximation algorithm is technically very different. Moreover, when the underlying graph is a directed layered graph (a setting similar to the one obtained in Asymmetric k-Center), the integrality gap of the LP-relaxation for RMFC becomes $\Omega(\log n)$, thus ruling out the possibility of extending our approach to this setting.

We note that independently of our work, Anshelevich et. al. [1] studied RMFC and related problems. In particular, they give an $O(\sqrt{n})$-approximation LP-rounding algorithm for RMFC on general graphs. They also show that the integrality gap of the LP they are using is $\Omega(\log n)$, and give an $O(\log \ell)$-approximation for directed layered graphs with ℓ layers. For the Firefighter problem, [1] prove an $n^{1-\epsilon}$-hardness of approximation, for any $0 < \epsilon < 1$. Additionally, Anshelevich et. al. study the same problems in the *spreading* model: when the problem is viewed in the context of containing a disease via vaccination, the spreading model assumes that the vaccination itself is an infectious process, whose spreading mechanism is similar to that of the infection itself. Anshelevich et. al. provide an $e/(e-1)$-approximation algorithm for the Firefighters problem, and an $O(\log n)$-approximation algorithm for RMFC in the spreading model. They also prove that RMFC in the spreading setting is at least as hard to approximate as the Set Cover problem.

Related Work The model for fire spread control was introduced by Hartnell [8], who studied the question of fire containment on infinite graphs. Under his definition, given an infinite graph G, we say that the fire is contained iff only a finite number of vertices burn. Wang and Moeller [20] showed that two firefighters are sufficient to contain the fire on a $\mathbb{Z}\times\mathbb{Z}$ grid, and that any algorithm using two firefighters needs at least 8 steps do so. Develin and Hartke [5] later showed that at least 18 vertices must burn in this scenario. Wang and Moeller [20] showed that $(r-1)$ firefighters are sufficient to contain the fire in any r-regular graph, and in particular this implies that $(2d-1)$ firefighters can contain the fire on a d-dimensional grid. Develin and Hartke [5] proved that this bound is tight, showing a lower bound of $(2d-1)$ firefighters for d-dimensional grids. In a more general setting, where the fire starts simultaneously at a finite number of vertices, Fogarty [7] showed that two firefighters are still sufficient to contain the fire on a 2-dimensional grid. In contrast, Develin and Hartke [5] proved that for d-dimensional grids, with $d \geq 3$, for any integer f, there is a finite set of fire outbreak locations for which f firefighters are insufficient.

Our Results and Techniques Our main result is an $O(\log^* n)$-approximation algorithm for RMFC on trees. The algorithm rounds a natural LP-relaxation of the problem. We then prove that an even stronger LP-relaxation has an $\Omega(\log^* n)$ integrality gap. We also consider RMFC on directed layered graphs and show an $O(\log n)$-approximation LP-rounding algorithm, matching the integrality gap of the LP relaxation.

We now provide a high level overview of the ideas and techniques used in our $O(\log^* n)$-approximation algorithm for RMFC on trees. We root the tree at the vertex s where the fire starts, and say that vertex v lies in layer τ iff its distance from s is τ. It is easy to see that we can assume w.l.o.g. that any solution chooses to save, at each time step τ, a subset of vertices lying in layer τ. With this observation, a natural LP relaxation for the problem assigns LP-weight $x(v)$ to each vertex v, with the constraints that the total LP-weight of vertices in any layer is at most k, the total LP-weight of any root-to-terminal path is at least 1, and the objective function of minimizing k. The main technical part of our algorithm is a randomized procedure, which, given an instance of RMFC on a tree and a set T of terminals, produces an almost-feasible near-optimal solution, in the following sense: the output consists of a collection of subsets U_τ of vertices to be saved at each time step τ, with $|U_\tau| \leq O(k)$ for each τ, and an additional set R of at most $\operatorname{poly}\log|T|$ vertices, such that sets $\{U_\tau\}_\tau$ induce a feasible solution in the graph obtained by removing the vertices of R from G. In other words, if we recursively obtain a feasible solution for the instance where the vertices of R serve as terminals, then the union of this solution with the collection $\{U_\tau\}_\tau$ of sets induces a feasible solution to the original problem. It is then easy to get an $O(\log^* n)$ approximation using the above procedure as a subroutine. The input graph G is partitioned into $g = O(\log^* n)$ sub-instances $H_1, H_2, \ldots, H_g$, such that each source-to-terminal path traverses every sub-instance in this order and has total LP-weight of at least $\Omega(1/g)$ inside each of them. We then apply the randomized rounding algorithm to tree H_g with the original set T of terminals, obtaining the sets $\{U^g_\tau\}_\tau$ and R^g, where $|R^g| \leq \operatorname{poly}\log|T|$. Set R^g of vertices is then used to define a set T_{g-1} of terminals for instance H_{g-1}, where each vertex $v \in R^g$ has an ancestor in T_{g-1}. The randomized rounding procedure is then applied

to (H_{g-1}, T_{g-1}), obtaining sets $\left(\left\{U_\tau^{g-1}\right\}_\tau, R^{g-1}\right)$, and so on. Since the size of the terminal set decreases fast with each iteration, with $|T_{i-1}| \leq \operatorname{poly}\log|T_i|$, set T_1 only contains a constant number of terminals, which are then added to the solution, together with sets $\left\{\bigcup_{h=1}^{g} U_\tau^h\right\}_\tau$. We note that in general, once the terminal set T_h becomes small, say $|T_h| \leq O(\log\log n)$, the problem can be solved efficiently via exhaustive search. However, the vertices of T_h do not necessarily belong to the original set T of terminals, and so we are not guaranteed that there is a near-optimal solution for the instance in which T_h is the set of terminals. Our LP-rounding algorithm proves that this indeed is the case, for the approximation factor of $O(\log^* n)$, while also providing a direct algorithm to find such a solution.

We now proceed to describe the randomized rounding procedure and its analysis, the main technical part of the paper. Observe first that the standard randomized rounding algorithm gives an $O(\log n)$-approximate solution for RMFC on trees, as follows: scan the layers from first to last, randomly selecting k vertices in each layer, with probabilities proportional to their LP-values. It is easy to show that this process disconnects a constant fraction of terminals from the root with high probability. Repeating this procedure $O(\log n)$ times gives a feasible $O(\log n)$-approximate solution. This scheme is however somewhat wasteful: if vertex v is selected to be in the solution, then there is no need to apply randomized rounding to its descendants, as the fire cannot reach any terminal in its subtree. Instead we can transfer the LP-weight from the descendants of v to other vertices, thus increasing the probability of choosing the remaining vertices and hence covering their descendants by the randomized rounding procedure. The first step of our algorithm is transforming the solution into a $1/M$ integral one, for $M = O(\log|T|)$, by a standard randomized rounding procedure. We then process the layers of the tree from first to last. At each step of the algorithm, we say that vertex u survives iff none of its ancestors has been added to the solution so far. Consider some surviving vertex v in some layer L_τ with positive LP-weight of, say, $x_v = 1/M$. Assume that the total LP-weight on the path connecting v to the root is h/M. Then we select v to be in the solution with probability roughly $(h+1)/M$. The resulting solution is guaranteed to be feasible, since for any path P connecting a terminal to the root, at least one vertex of P is in the solution (in particular, the last vertex of P with non-zero LP-value is added to the solution with probability 1 if none of its ancestors belongs to it). The main technical part of the analysis is to show that the cost of the obtained solution is $O(k)$ with high probability. This is done by introducing a weight transfer mechanism: a way to transfer LP-weight from vertices whose ancestors are already in the solution to the remaining vertices. The weight transfer mechanism must on one hand ensure that the LP weight of every layer does not increase, while on the other hand, the LP-weight of each surviving vertex grows fast enough. We are unable to find a perfect weight transfer mechanism (in fact the lower bound on the integrality gap shows that it does not exist). Instead we add, throughout the algorithm, ancestors of vertices whose LP weight does not grow fast enough, to the set R. The heart of the analysis is designing a weight transfer scheme with the desired properties, while ensuring that $|R| \leq \operatorname{poly}\log|T|$.

Organization: We provide an $O(\log^* n)$-approximation algorithm for RMFC on trees in Section 3 and prove a matching lower bound on the integrality gap of the stronger LP in Section 6. Section 7 contains an $O(\log n)$-approximation algorithm for directed layered graphs. For completeness, we sketch the matching lower bound of [12] on the integrality gap of the LP in Appendix.

2 Preliminaries

In the RMFC problem, we are given an undirected graph $G = (V, E)$, a source vertex $s \in V$ where the fire starts, and a subset $T \subseteq V$ of terminals that need to be protected. At each time step τ, we need to choose a subset U_τ of at most k vertices to be saved. Once a vertex burns or is saved, it remains in this state permanently. Let B_τ denote the set of burning vertices at time step τ, with $B_0 = \{s\}$. Vertex v burns at time $\tau+1$ iff it has not been saved so far, that is, $v \notin \bigcup_{1\leq\tau'\leq\tau+1} U_{\tau'}$, and at least one neighbor of v is in B_τ. Our goal is to select subsets $\{U_\tau\}_{\tau=1}^n$ of vertices to be saved at each time step τ, so that the fire does not spread to the terminals, while minimizing the maximum number of vertices to be saved at any time step, $k = \max_\tau\{|U_\tau|\}$. Formally, given a solution $\mathcal{S} = \{U_\tau\}_{\tau=1}^n$, we say that a simple path $P = (s, v_1, v_2, \ldots, v_z)$ is a *fire spreading path* iff for each $\tau : 1 \leq \tau \leq z$, $v_\tau \notin \bigcup_{\tau'\leq\tau} U_{\tau'}$. We say that vertex v burns in solution $\mathcal{S}$ iff it lies on any fire spreading path. Otherwise, we say that v is *protected* by $\mathcal{S}$. A solution is feasible iff all terminals are protected. We consider a natural special case of RMFC where G is a tree rooted at s. Let L_i denote the ith layer of the tree, containing all vertices whose distance from s is i, with $L_0 = \{s\}$, and L_λ being the last layer.

OBSERVATION 2.1. *Let $\{U_\tau\}_{\tau=1}^n$ be any feasible solution to the RMFC problem on tree G. Then we can assume, w.l.o.g., that for every $\tau : 1 \leq \tau \leq \lambda$, $U_\tau \subseteq L_\tau$, and $U_\tau = \emptyset$ for all $\tau > \lambda$.*

Proof. Assume otherwise. Consider some set U_τ, and let $v \in U_\tau \setminus L_\tau$ be any vertex. Assume that $v \in L_{\tau'}$. If $\tau' > \tau$, then let u be the ancestor of v in L_τ. We replace v with u in U_τ. Clearly, this does not change the solution cost, and the solution remains feasible. Assume now that $\tau' < \tau$. Then we simply remove v from U_τ. It is easy to see that the solution remains feasible: if v is not protected by the solution, then it starts burning at step τ', so saving v at step $\tau > \tau'$ does not affect the spread of the fire and the feasibility of the solution. □

For a vertex $v \in V$, let P_v denote the path connecting v to the root s. Let $U \subseteq V$ be any subset of vertices. We say that U is a *feasible set* iff for every terminal $t \in T$, $U \cap P_t \neq \emptyset$. Given such set U, we define $U_\tau = U \cap L_\tau$ to be the subset of vertices to be saved at time step τ. Recall that the cost of the solution $\mathcal{S} = \{U_\tau\}_{\tau=1}^{\lambda}$ is $\max_\tau \{|U_\tau|\}$. It will be more convenient to work with a slightly different notion of solution cost, called *amortized cost*. We say that a set U of vertices has amortized cost k' iff for every $1 \leq \tau \leq \lambda$, $\sum_{\tau'=1}^{\tau} |U_{\tau'}| \leq k'\tau$, where $U_\tau = U \cap L_\tau$. Clearly, the amortized cost of the optimal feasible solution is at most OPT. The next claim shows that any feasible set with amortized cost k' can be converted into a feasible solution of (non-amortized) cost k'. RMFC is therefore equivalent to the problem of finding a feasible set U of minimum amortized cost.

Claim 2.1. *Given a feasible set U of amortized cost k', we can efficiently find a feasible solution $\{U'_\tau\}_{\tau=1}^{\lambda}$ of (non-amortized) cost at most k'.*

Proof. For each $\tau : 1 \leq \tau \leq \lambda$, let $U_\tau = U \cap L_\tau$. Scan sets $\{U_\tau\}_{\tau=1}^{\lambda}$ in the decreasing order of τ. Let U_τ be the current set, and assume that $|U_\tau| = k'' > k'$. Choose an arbitrary subset $S \subseteq U_\tau$ of $k'' - k'$ vertices. Remove vertices of S from U_τ, and for each $v \in S$, add the ancestor of v in layer $(\tau - 1)$ to $U_{\tau-1}$. It is easy to verify that the amortized cost of the new solution does not increase and the solution remains feasible. After all subsets U_τ are processed in this manner, the number of vertices in each subset is at most k'. □

The LP Relaxation. For each vertex v, we have an indicator variable $x(v)$ for including v in the solution. Consider the following LP relaxation:

$$\begin{array}{lccl} \text{(LP)} & \min & k & \\ \text{s.t.} & \sum_{v \in L_\tau} x(v) \leq k & \forall \tau : 1 \leq \tau \leq \lambda \\ & \sum_{v \in P_t} x(v) \geq 1 & \forall t \in T \\ & x(v) \geq 0 & \forall v \in V \end{array}$$

The first set of constraints bounds the number of vertices chosen in each layer of the tree by k, the solution value, while the second set ensures that for every terminal $t \in T$, at least one vertex on path P_t is saved. We use the regular notion of solution cost in the LP relaxation, while we will use the amortized cost in LP-rounding. Notice that the optimal solution cost $\mathsf{OPT} \geq \lceil k \rceil$.

3 The Algorithm

We use a parameter $g = O(\log^* n)$, and our final approximation factor is $O(g)$. We can assume, w.l.o.g., that $x(v) \leq 1/(2g)$: otherwise, we can add all vertices v with $x(v) > 1/2g$ to the solution, remove their subtrees from G and solve the remaining instance. This will only increase the approximation ratio by at most a factor of 2. Our first step is to partition the tree G into a collection of g subgraphs $H_1, \ldots, H_g$, such that any root-to-terminal path in G traverses $H_1, \ldots, H_g$ in this order and has an LP-weight of roughly $1/g$ inside each. First, we partition the set V of vertices into g subsets $V_1, \ldots, V_g$, as follows. Add s to V_1 and scan the remaining vertices in the non-decreasing order of their distance from s, breaking ties arbitrarily. Let v be the current vertex, and assume that the parent of v belongs to V_j. If $\sum_{u \in P_v \cap V_j} x(u) < 1/(2g)$, then v is added to V_j, otherwise, it is added to V_{j+1}. It is easy to see that every root-to-terminal path P in G traverses $V_1, V_2, \ldots, V_g$ in this order, and for all j, $1/(2g) \leq \sum_{u \in P \cap V_j} x(u) \leq 1/g$. To construct graph H_j, we take the subgraph of G induced by V_j, and if $j > 1$, we turn it into a tree by adding a source vertex s_j that connects to every vertex $v \in V_j$ whose parent does not belong to V_j. We think for now of the leaves of H_j as being the terminals in the corresponding RMFC instance. The LP-solution x_j associated with H_j is defined as follows: For $v \in V_j$, $x_j(v) = 2gx(v)$. Notice that the LP-weight of any root-to-leaf path inside H_j is at least 1.

We will be solving each instance H_j separately. Since H_j is a tree, it can also be viewed as a layered graph. The layering of vertices in H_j may be different from the original one, e.g. the first layer of H_j may contain vertices from various subsets L_τ. However, while solving instance H_j, we need to keep track of the original layering of its vertices. This motivates the notation we now introduce. Let X_τ^j denote the total LP-weight of vertices in L_τ in the LP-solution for H_j, $X_\tau^j = \sum_{v \in L_\tau \cap V_j} x_j(v)$. Recall that for each layer L_τ, $\sum_{j=1}^{g} X_\tau^j \leq 2gk$, due to the scaling of LP-values $x(v)$ by the factor of $2g$. Given any vertex $v \in V_j$, let P_v^j denote the path connecting v to s_j in graph H_j. Let T_j be any subset of leaves of the tree H_j. We say that a subset

$U^j \subseteq V_j$ of vertices is a feasible solution for instance (H_j, T_j) iff for each $t \in T_j$, at least one vertex on path P_t^j belongs to U^j. We define, for each $1 \leq \tau \leq \lambda$, $U_\tau^j = U^j \cap L_\tau$ to be the subset of vertices of U^j contained in layer τ of G.

We are now ready to present the high-level idea of our algorithm. We start by solving the sub-instance (H_g, T_g), where T_g is a subset of leaves of H_g, containing one ancestor for each terminal $t \in T$. Our goal is to produce a feasible solution U^g, where for each $1 \leq \tau \leq \lambda$, $|U_\tau^g| \leq O(X_\tau^g)$ (in fact we will be using amortized costs). Instead of finding such a solution, we will find a *partial solution*: sets U^g, R^g such that $U^g \cup R^g$ is a feasible solution for (H_g, T_g), $|R^g| \leq \text{poly}\log|T_g|$, and the amortized cost of solution U^g is low. While $|R^g|$ is relatively small, it is possible that many vertices of R^g lie in the same layer L_τ, while $|R^g| >> k$, so we cannot output $U^g \cup R^g$ as the final solution, as its cost may be too high. Instead, we consider the instance H_{g-1}, together with terminal set T_{g-1}, containing, for each vertex $v \in R^g$, the unique leaf of H_{g-1} lying on P_v. We then continue the same process on H_{g-1}, obtaining an even smaller set R^{g-1} and so on. The main technical part of the algorithm is summarized in the following theorem.

THEOREM 3.1. *For each $1 \leq j \leq g$, for each subset T_j of leaves of H_j, there is an efficient algorithm for finding sets $U^j, R^j \subseteq V_j$ of vertices, such that:*

- $|R^j| \leq O(\log^5 |T_j|)$.
- *$U^j \cup R^j$ is a feasible solution for instance (H_j, T_j), that is, for each $t \in T_j$, $P_t^j \cap \left(R^j \cup U^j\right) \neq \emptyset$.*
- *For all $1 \leq \tau \leq \lambda$, $\sum_{\tau'=1}^{\tau} |U_{\tau'}^j| \leq O(\tau + \sum_{\tau'=1}^{\tau} X_{\tau'}^j)$ w.h.p., where $U_\tau^j = U^j \cap L_\tau$.*

We defer the proof of this theorem to the next section. We now show that it implies an $O(g)$-approximation algorithm. Apply Theorem 3.1 to graphs $H_g, H_{g-1}, \ldots, H_1$ in this order. In iteration i, Theorem 3.1 is applied to graph H_{g-i+1} together with terminal set T_{g-i+1}, computed as follows. For $i = 1$, T_g contains, for each terminal $t \in T$, the unique leaf of H_g lying on P_t. For $i > 1$, let R^{g-i+2} be the set in the output of Theorem 3.1, computed in iteration $(i-1)$. For each vertex $v \in R^{g-i+2}$, let v' be the unique leaf of tree H_{g-i+1}, lying on path P_v in G. We then add v' to T_{g-i+1}. Therefore, $|T_{g-i+1}| \leq |R^{g-i+2}| \leq O(\log^5 |T_{g-i+2}|)$. We now use the following simple claim, whose proof appears in Appendix.

CLAIM 3.1. *For a suitably chosen $g = O(\log^* n)$, $|T_1|$ is bounded by a constant.*

The final solution is $U = \left(\bigcup_{j=1}^{g} U^j\right) \cup T_1$. For each $\tau : 1 \leq \tau \leq \lambda$, let $U_\tau = U \cap L_\tau$. Theorem 3.1 ensures that $\sum_{\tau' \leq \tau} |U_{\tau'}| = \sum_{j=1}^{g} \sum_{\tau' \leq \tau} |U_{\tau'}^j| + |T_1| \leq \sum_{j=1}^{g} O\left(\tau + \sum_{\tau' \leq \tau} X_{\tau'}^j\right) + O(1) \leq O(g\tau) + O(g\tau \mathsf{OPT}) + O(1) = O(g\tau\mathsf{OPT})$. Therefore, the amortized cost of the solution U is $O(g \cdot \mathsf{OPT})$. It is easy to see that U is a feasible set, as every terminal t has at least one vertex $v \in P_t \cap U$.

4 Proof of Theorem 3.1

We start with the tree H_j and a subset T_j of leaves of H_j, that we call terminals. Since from now on we focus on a specific tree H_j, to simplify the notation we will omit the index j. So we denote H_j by H, V_j by V, and for all $1 \leq \tau \leq \lambda$, $L_\tau^j = V_j \cap L_\tau$ is denoted by $\mathcal{L}_\tau$. We denote T_j by $\mathcal{T}$, and the total number of terminals is denoted by $N = |\mathcal{T}|$. The LP-solution $x_j(v)$ is denoted by $x(v)$ and X_τ^j is denoted by X_τ, so $X_\tau = \sum_{v \in \mathcal{L}_\tau} x(v)$. We also use the fractional amortized cost, $\overline{X}_\tau = \sum_{\tau' \leq \tau} X_{\tau'}$ in our analysis. For a vertex $v \in V$, let P_v be the unique path connecting v to the root s of H. Recall that for each $t \in \mathcal{T}$, $\sum_{v \in P_t} x(v) \geq 1$. Our goal is to find two sets $U, R \subseteq V$ of vertices, such that $|R| = O(\log^5 N)$, and for each terminal $t \in \mathcal{T}$, $P_t \cap (R \cup U) \neq \emptyset$. For each $1 \leq \tau \leq \lambda$, denote $U_\tau = U \cap \mathcal{L}_\tau$. We also need to ensure that the amortized cost $\sum_{\tau'=1}^{\tau} |U_{\tau'}| \leq O(\tau + \overline{X}_\tau)$ for all τ. We use a parameter M, the smallest power of 2 whose value is greater than $100 \log N$, so $M = \Theta(\log N)$. The algorithm has three steps. In the first step we transform the fractional solution into a $1/M$-integral one, with only $O(N^2)$ vertices having non-zero LP-weight.

In the second step we partition the vertices with non-zero LP-weight into subsets $F_1, F_2, \ldots, F_M$. In the last step we compute the set R and perform randomized rounding to obtain set U.

Step 1: Obtaining a $1/M$-integral solution Given any fractional solution y, let $Y_\tau = \sum_{v \in \mathcal{L}_\tau} y(v)$ and $\overline{Y}_\tau = \sum_{\tau' \leq \tau} Y_{\tau'}$ for all $1 \leq \tau \leq \lambda$. This step is summarized in the next theorem whose proof uses standard randomized rounding techniques and appears in Appendix.

THEOREM 4.1. *Given any feasible fractional solution x for instance $(H, \mathcal{T})$, there is an efficient randomized algorithm to find a $(1/M)$-integral feasible solution y, such that the number of vertices v with $y(v) > 0$ is at most N^2, and for all $\tau \geq 1$, $\overline{Y}_\tau \leq O(\tau + \overline{X}_\tau)$. The algorithm suceeds with probability at least $1 - 2/N^2$.*

Let $\mathcal{E}_1$ be the event that the algorithm in Theorem 4.1 succeeds, so $\mathbf{Pr}[\mathcal{E}_1] \geq 1 - 2/N^2$.

Step 2: Defining Sets $F_1, \ldots, F_M$. Let $v \in V$ be any vertex with $y(v) = r/M$ for some integer $r > 0$, and assume that $\sum_{u \in P_v \setminus \{v\}} y(u) = h/M$, for some integer $h \geq 0$. Then v belongs to sets $F_{h+1}, F_{h+2}, \ldots, F_{h+r}$. We view the weight $y(v)$ of v as being evenly split among the r sets, and so the weight of v with respect to $F_{h+r'}$ is $y_{h+r'}(v) = 1/M$, for all $1 \leq r' \leq r$. We then have that $\sum_{h: v \in F_h} y_h(v) = y(v)$. For each terminal $t \in \mathcal{T}$, for each $h : 1 \leq h \leq M$, there is exactly one vertex $v \in P_t \cap F_h$. We now group sets $F_1, \ldots, F_M$ geometrically, by defining $\log M$ sets of indices $I_1, \ldots, I_{\log M}$ (recall that M is a power of 2). Set I_1 contains the first $M/2$ indices, set I_2 contains the next $M/4$ indices, and so on, with the last set $I_{\log M}$ containing a single index M. Formally: $I_q = \left\{ b + r \mid b = \sum_{1 \leq q' < q} M/2^{q'}, 1 \leq r \leq M/2^q \right\}$.

Step 3: Randomized Rounding This is the main step of the algorithm. The goal is to choose a set U of vertices that will serve as the output. We will also define a set R of vertices, $R = \bigcup_{h=0}^{M} R_h$. The algorithm has M iterations. Set R_0 is selected before the first iteration starts (we specify its choice below), and for each $h : 1 \leq h \leq M$, set R_h is selected in iteration h.

Consider the beginning of iteration h and assume that $h \in I_q$ for some $1 \leq q \leq \log M$. Vertex $v \in F_h$ is called *active* iff no vertex of P_v currently belongs to $R \cup U$. Let $A_h \subseteq F_h$ denote the subset of vertices of F_h that are active at the beginning of iteration h. We select a subset $R_h \subseteq A_h$ of vertices to be added to R (we show how to select this subset below). Next, each vertex $v \in A_h \setminus R_h$ is added to U with probability $\min\{16 \cdot 2^q/M, 1\}$.

This completes the description of the algorithm, except for the definition of the sets R_h. It will be convenient to partition the algorithm's execution into $\log M$ phases, where phase q consists of iterations $h \in I_q$. We now define $U_\tau = U \cap \mathcal{L}_\tau$ for all $1 \leq \tau \leq \lambda$. It is easy to see that $U \cup R$ is a feasible solution for $(H, \mathcal{T})$, since for every terminal $t \in \mathcal{T}$, $P_t \cap (U \cup R) \neq \emptyset$ (otherwise, the vertex $v \in P_t \cap F_M$ should have been added to U with probability 1).

It is easy to see that the expected amortized cost of solution U for each τ is low. In particular, if we show that each vertex $v \in F_h$ is added to U with probability at most $O(y_h(v))$, then the expected amortized cost of U for each τ is $O(\overline{Y}_\tau)$: Consider some $v \in F_h$ for some $h \in I_q$. By the definition of the decomposition $F_1, F_2, \ldots, F_M$, vertex v has one ancestor $v_{h'} \in F_{h'}$ for all $h' < h$. Vertex v is added to set U iff it remains active until iteration h, and it is chosen by the randomized rounding procedure. Consider some $q' < q$. The probability that no vertex in set $\{v_{h'} : h' \in I_{q'}\}$ is added to U is bounded by $(1 - 2^{q'+4}/M)^{M/2^{q'}} \leq 1/2$, so the probability that v remains active at the beginning of iteration h is at most $1/2^{q-1}$. Since v is selected by the randomized rounding procedure with probability at most $16 \cdot 2^q/M$, overall v is added to U with probability at most $O(y_h(v)) = O(1/M)$. Therefore, the expected amortized cost of U for every τ is $O(\overline{Y}_\tau)$. This bound on the expectation is however not enough for us, and is only provided here for intuition. We need to prove that the amortized cost of the set U is low for each τ with high probability; the proof of this claim is more involved.

Analysis. Recall that if event $\mathcal{E}_1$ happens, then $\overline{Y}_\tau \leq O(\tau + \overline{X}_\tau)$ for all $1 \leq \tau \leq \lambda$. In order to bound the amortized solution cost, it is therefore enough to prove that with high probability:

$$\forall \tau : 1 \leq \tau \leq \lambda \quad \sum_{\tau' \leq \tau} |U_{\tau'}| \leq O(\overline{Y}_\tau) \tag{4.1}$$

Recall that the partition $\{\mathcal{L}_\tau\}_{\tau=1}^{\lambda}$ of vertices of H corresponds to the original layers in graph G, while subsets $\{F_h\}_{h=1}^{M}$ are defined according to the distribution of the LP-weight on vertices of H. We break the summation in (4.1) down by sets $F_1, \ldots, F_M$ as follows. For each $1 \leq \tau \leq \lambda$, for each $1 \leq h \leq M$, let $\mathcal{L}_{\tau,h} = F_h \cap \mathcal{L}_\tau$ and $U_{\tau,h} = F_h \cap U_\tau$. Let $Y_{\tau,h} = \sum_{v \in \mathcal{L}_{\tau,h}} y_h(v)$ and $\overline{Y}_{\tau,h} = \sum_{\tau' \leq \tau} Y_{\tau',h}$. So $\sum_{h=1}^{M} \overline{Y}_{\tau,h} \leq \overline{Y}_\tau$. In order to show that (4.1) holds w.h.p., it is now enough to show that w.h.p.:

$$\forall h : 1 \leq h \leq M, \forall \tau : 1 \leq \tau \leq \lambda \quad \sum_{\tau' \leq \tau} |U_{\tau',h}| \leq 200 \overline{Y}_{\tau,h} \tag{4.2}$$

For each $\tau : 1 \leq \tau \leq \lambda$, let $A_{\tau,h}$ contain the vertices of $F_h \cap \mathcal{L}_\tau$, that are active at the beginning of iteration h, i.e. $A_{\tau,h} = A_h \cap \mathcal{L}_\tau$. We say that event $\mathcal{E}_2(h)$ holds for $h \in I_q$ iff:

$$\forall \tau : 1 \leq \tau \leq \lambda \quad \sum_{\tau' \leq \tau} |A_{\tau',h}| \cdot \frac{2^q}{M} \leq 2\overline{Y}_{\tau,h} \tag{4.3}$$

Let $\mathcal{E}_2 = \bigcap_{1 \leq h \leq M} \mathcal{E}_2(h)$. The heart of our algorithm and its analysis is to show the choice of sets R_h, for which events $\mathcal{E}_2(h)$ happen with high probability for all h. The formal statement is summarized in the following theorem:

THEOREM 4.2. *There is an efficient procedure for selecting, in each iteration $h' : 1 \leq h' \leq M$, a subset $R_{h'} \subseteq A_{h'}$ of vertices, with $|R_{h'}| \leq O(\log^4 N)$, such*

that for each $h : 1 \leq h \leq M$, event $\mathcal{E}_2(h)$ happens with probability at least $1 - 1/N^2$.

The proof of Theorem 4.2 appears in the next section. We first complete the proof of Theorem 3.1 using Theorem 4.2. If event $\mathcal{E}_2(h)$ happens, then $\sum_{\tau' \leq \tau} |A_{\tau',h}| \cdot \frac{2^q}{M} \leq 2\overline{Y}_{\tau,h}$. In iteration h, our algorithm chooses, for each τ, vertices in $A_{\tau,h} \setminus R_h$ to be added to $U_{\tau,h}$ with probability $\min\{16 \cdot 2^q/M, 1\}$ each. So intuitively, if (4.3) holds, then from Chernoff bound, (4.2) also holds with high probability for all τ and h. This approach indeed works, except for the cases where $\overline{Y}_{\tau,h}$ is small, and so the probability of success guaranteed by the Chernoff bound is low. We take care of this problem by adding vertices to set R_0 as follows. Fix some $h : 1 \leq h \leq M$. Let τ_h be the largest index for which $\overline{Y}_{\tau_h,h} \leq M$. Add all the vertices in $\bigcup_{\tau \leq \tau_h} \mathcal{L}_{\tau,h}$ to R_0. Notice that since every vertex in F_h has LP-value $y_h(v) \geq 1/M$, we add at most M^2 vertices for each h, so $|R_0| \leq M^3$. It is now easy to complete the proof of Theorem 3.1. Let $\mathcal{E}_3(h)$ be the event that (4.2) holds for h, and let $\mathcal{E}_3 = \bigcap_{1 \leq h \leq M} \mathcal{E}_3(h)$. The proof of the next theorem follows from applying standard Chernoff bounds.

THEOREM 4.3. *Event $\mathcal{E}_3$ happens with probability at least $1 - 1/N$.*

Proof. We start with the following claim:

CLAIM 4.1. *For each $h : 1 \leq h \leq M$, we have that $\mathbf{Pr}\left[\neg\mathcal{E}_3(h) \mid \mathcal{E}_1 \wedge \mathcal{E}_2(h)\right] \leq 1/N^2$.*

Proof. Assume that $h \in I_q$ and consider some $\tau : 1 \leq \tau \leq \lambda$. If $\tau \leq \tau_h$, then $U_{\tau,h} = \emptyset$ for each $\tau' \leq \tau$, so Equation (4.2) holds for this choice of τ, h. Otherwise, if $\tau > \tau_h$, then since we condition on event $\mathcal{E}_2(h)$, $\sum_{\tau' \leq \tau} |A_{\tau',h}| \cdot \frac{2^q}{M} \leq 2\overline{Y}_{\tau,h}$. For each $v \in \left(\bigcup_{\tau' \leq \tau} A_{\tau',h}\right) \setminus R_h$, let Z_v be the indicator variable for including v in U, so $\sum_{\tau' \leq \tau} |U_{\tau',h}| = \sum_{\tau' \leq \tau} \sum_{v \in A_{\tau',h} \setminus R_h} Z_v$. Then $\mathbf{E}\left[\sum_{\tau' \leq \tau} \sum_{v \in A_{\tau',h} \setminus R_h} Z_v\right] \leq 16 \sum_{\tau' \leq \tau} |A_{\tau',h}| \cdot \frac{2^q}{M} \leq 32\overline{Y}_{\tau,h}$. Using the Chernoff bound, $\mathbf{Pr}\left[\sum_{\tau' \leq \tau} |U_{\tau',h}| > 200\overline{Y}_{\tau,h}\right] \leq 2^{-200\overline{Y}_{\tau,h}} \leq 2^{-200M}$, since $\overline{Y}_{\tau,h} \geq M$. Applying the union bound over all values $\tau : \tau_h \leq \tau \leq \lambda$ for which $\mathcal{L}_{\tau,h} \neq \emptyset$ (notice that there are at most $N^2 \cdot M$ such values), we get the desired result. □

Since $\mathbf{Pr}\left[\neg\mathcal{E}_3(h)\right] \leq \mathbf{Pr}\left[\neg\mathcal{E}_3(h) \mid \mathcal{E}_1 \wedge \mathcal{E}_2(h)\right] + \mathbf{Pr}\left[\neg(\mathcal{E}_1 \wedge \mathcal{E}_2(h))\right] \leq \mathbf{Pr}\left[\neg\mathcal{E}_3(h) \mid \mathcal{E}_1 \wedge \mathcal{E}_2(h)\right] + 3/N^2$, we get that $\mathbf{Pr}\left[\neg\mathcal{E}_3(h)\right] \leq 4/N^2$. Using the union bound over all indices $h : 1 \leq h \leq M$ finishes the proof. □

Finally, we set $R = \bigcup_{h=0}^{M} R_h$. Since $|R_h| \leq O(\log^4 N)$ for all h, we have that $|R| \leq O(\log^5 N)$ as required. This completes the proof of Theorem 3.1, except for the proof of Theorem 4.2.

5 Proof of Theorem 4.2

We start with a high-level overview of the proof. For a vertex $v \in H$, let $\mathbf{L}(v)$ be the layer $\mathcal{L}_\tau$ to which v belongs. Consider the set F_h for some fixed $h \in I_q$, for some $1 \leq q \leq \log M$, and the partition $F_h = \bigcup_{\tau=1}^{\lambda} \mathcal{L}_{\tau,h}$ of set F_h into layers, where $\mathcal{L}_{\tau,h} = F_h \cap \mathcal{L}_\tau$. We say that $v \in F_h$ is *active* at the beginning of iteration $h' < h$ iff no vertex of P_v has been added to $U \cup R$ in iterations $1, 2, \ldots, h' - 1$. Let $A_h^{h'} \subseteq F_h$ denote the set of active vertices of F_h at the beginning of iteration h', so $A_h^1 = F_h \setminus R_0$, and let $A_{\tau,h}^{h'} = A_h^{h'} \cap \mathcal{L}_\tau$. Similarly, for each phase $q' : 1 \leq q' \leq q$, we denote by $A_h^{(q')} \subseteq F_h$ the set of vertices that are active at the beginning of phase q'. Since the LP-weight of each vertex $v \in F_h$ w.r.t. F_h is $y_h(v) = \frac{1}{M}$, we have, for each $\tau : 1 \leq \tau \leq \lambda$, $|A_{\tau,h}^1| \cdot \frac{1}{M} \leq Y_{\tau,h}$, and $\sum_{\tau' \leq \tau} |A_{\tau',h}^1| \cdot \frac{1}{M} \leq \overline{Y}_{\tau,h}$ (the inequality is since $A_{\tau,h}^1$ does not include the vertices of R_0, which are counted in $\overline{Y}_{\tau,h}$). As the algorithm progresses, some of the vertices of $F_h \setminus R_0$ become inactive. Our main idea is to transfer the LP-weight from such vertices to the remaining active vertices of F_h. The transfer is only performed from vertex $v \in \mathcal{L}_\tau$ to vertex $v' \in \mathcal{L}_{\tau'}$ if $\tau \leq \tau'$. This ensures that throughout the algorithm, the amortized fractional weight $\overline{Y}_{\tau,h}$ does not increase for any τ. For each vertex $v \in A_h^{h'}$, we denote by $y_h^{h'}(v)$ its LP-weight at the beginning of iteration h', and by $y_h^{(q')}(v)$ its LP-weight at the beginning of phase q'. Our goal is to ensure that for each phase $1 \leq q' \leq q$, for each vertex $v \in A_h^{(q')}$ that remains active at the beginning of phase q', its LP-weight $y_h^{(q')}(v) \geq \frac{2^{q'-1}}{M}$. Therefore, the LP-weight of the surviving active vertices doubles in each phase, and eventually, at the beginning of iteration h, $y_h^h(v) \geq \frac{2^{q-1}}{M}$ for each active vertex v. Since the amortized fractional LP-weight $\overline{Y}_{\tau,h}$ does not increase throughout the algorithm, this will give the desired bound $\sum_{\tau' \leq \tau} |A_{\tau',h}^h| \cdot \frac{2^q}{M} \leq 2\overline{Y}_{\tau,h}$ for all τ. The main ingredient of our analysis is a *weight transfer mechanism* for moving the LP-weight from vertices that become inactive to those that remain active in F_h.

We now provide an informal overview of the weight transfer scheme. We focus on the weight transfer within a specific set F_h, where $h \in I_q$ for some $1 \leq q \leq \log M$. Let $h' < h$ be the index of the current iteration, and $h' \in I_{q'}$. Assume that at the beginning of phase q', the LP-weight of each active vertex $v \in A_h^{(q')}$ is

$y_h^{(q')}(v) \geq \frac{2^{q'-1}}{M}$. We need to ensure that the LP-weight of each surviving active vertex of F_h becomes at least $2^{q'}/M$ at the end of phase q'. Since phase q' has $M/2^{q'}$ iterations, it is enough to ensure that the LP-weight of each surviving active vertex increases by at least $(2^{q'}/M)^2$ in each iteration $h' \in I_{q'}$. Consider the following simple weight transfer mechanism: Fix some partition of the active vertices $A_h^{h'} \subseteq F_h$ into subsets $Z_1, \ldots, Z_r$ of roughly equal size. For each subset $Z_j, 1 \leq j \leq r$, if some vertex $v \in Z_j$ becomes inactive during iteration h', then its LP-weight is evenly split among the vertices of Z_{j+1}. Observe that for each active vertex $v \in A_h^{h'}$, the probability to become inactive in iteration h' is at least $16 \cdot 2^{q'}/M$ (this is the probability that the ancestor of v in $F_{h'}$ is added to U). So we expect a $(16 \cdot 2^{q'}/M)$-fraction of vertices of Z_j to become inactive. In case a close number of vertices (say, $|Z_j| \cdot 8 \cdot 2^{q'}/M$ vertices) become inactive, each vertex in Z_{j+1} will receive a total contribution of $\frac{|Z_j| \cdot 8 \cdot 2^{q'}}{M} \cdot \frac{2^{q'-1}}{M} \cdot \frac{1}{|Z_{j+1}|} \geq (2^{q'}/M)^2$, as desired. There are two main obstacles to this approach. First, in order to show that w.h.p., for each Z_j, the number of vertices that become inactive is close to the expected one, we need to ensure that there is a certain degree of independence among these events (for example, if all vertices in Z_j have one common ancestor in $F_{h'}$, this will not be the case). So ideally, we would like set Z_j to contain a roughly equal number of descendants of many (say $\Omega(\log^2 N)$) distinct vertices of $A_{h'}^{h'}$. At the same time, in order to guarantee that the amortized cost does not grow, we need to ensure that for each $v \in Z_j$ and $u \in Z_{j+1}$, $\mathbf{L}(v) \leq \mathbf{L}(u)$ for all j, so the weight is only transferred from layer $\mathcal{L}_\tau$ to layer $\mathcal{L}_{\tau'}$ where $\tau \leq \tau'$. This motivates the weight transfer mechanism that we formally describe below.

We now turn to the formal proof of Theorem 4.2. The proof consists of two parts. In the first part we define the procedure for selecting subsets $R_{h'} \subseteq A_{h'}^{h'}$ of vertices to be added to the set R in each iteration h'. The second part proves that for each $h : 1 \leq h \leq M$, event $\mathcal{E}_2(h)$ happens with high probability, under this choice of sets $R_{h'}$. We start with the first part. Consider some iteration h' of the algorithm, $1 \leq h' \leq M$. For each $h : h' < h \leq M$, we define a partition $C_{i,j}^h$ of the set $A_{h'}^{h'}$ of vertices. (Notice that each $h > h'$ defines a distinct partition $\left\{C_{i,j}^h\right\}$ of $\mathcal{A}_{h'}^{h'}$.) This partition is used, on the one hand, to define a subset $R_{h',h} \subseteq A_{h'}^{h'}$ of vertices, and on the other hand, we later use it to define a weight transfer mechanism. Eventually we set $R_{h'} = \bigcup_{h>h'} R_{h',h}$.

Defining Sets $C_{i,j}^h$ and set $R_{h',h}$. We now fix some index $h : h' < h \leq M$ and define the partition $\left\{C_{i,j}^h\right\}$ of $A_{h'}^{h'}$ induced by F_h. To simplify the notation, we denote the sets $C_{i,j}^h$ by $C_{i,j}$ here.

Consider first some vertex $v \in A_{h'}^{h'}$ that is active at the beginning of iteration h'. Let $D(v) \subseteq A_h^{h'}$ be the set of the descendants of v in F_h that are currently active. We order the vertices $u \in D(v)$ in the non-decreasing order of layers $\mathbf{L}(u)$, $D(v) = \{u_1, \ldots, u_{z_v}\}$, where $\mathbf{L}(u_1) \leq \mathbf{L}(u_2) \leq \cdots \leq \mathbf{L}(u_{z_v})$. We now group the vertices of $D(v)$ geometrically, with $D_1(v) = \{u_1\}$, $D_2(v)$ containing the next two vertices and so on. Formally, for $1 \leq i \leq \lfloor \log z_v \rfloor$, $D_i(v) = \left\{u_{b+r} \mid 1 \leq r \leq 2^{i-1}, b = \sum_{i'<i} 2^{i'-1}\right\}$, and $D_{\lceil \log z_v \rceil}(v)$ contains the remaining vertices, $D_{\lceil \log z_v \rceil}(v) = D(v) \setminus \left(\bigcup_{i<\lceil \log z_v \rceil} D_i(v)\right)$.

We now define $\log N$ classes of vertices in $A_{h'}^{h'}$, as follows: $C_i = \left\{v \in A_{h'}^{h'} : |D_i(v)| = 2^{i-1}\right\}$ (notice that vertex v belongs to $\Theta(\log z_v)$ such classes). Observe that the set $\bigcup_{v \in C_i} D_i(v)$ contains exactly 2^{i-1} descendants for each vertex $v \in C_i$. We group the vertices of C_i into subsets, containing $O(\log^2 N)$ vertices each. The descendants $D_i(v)$ of vertices in each such subset will serve as the sets Z_j from the intuitive explanation above. Since we can only transfer LP-weight from layer $\mathcal{L}_\tau$ to layer $\mathcal{L}_{\tau'}$ where $\tau \leq \tau'$, we need to perform this grouping carefully.

Fix some class C_i, for $1 \leq i \leq \log N$. For each vertex $v \in C_i$, consider the corresponding set $D_i(v)$ of its 2^{i-1} descendants in $A_h^{h'}$. Let $\tau_i(v)$ be the largest index τ of a layer to which any vertex in $D_i(v)$ belongs, so $\tau_i(v) = \max_{u \in D_i(v)} \{\mathbf{L}(u)\}$. We now order the vertices of C_i in the non-decreasing order of $\tau_i(v)$, and partition them into consecutive subsets of $10M^2$ vertices each in this order. Formally, we partition set C_i into $\alpha_i = \left\lceil |C_i|/10M^2 \right\rceil$ subsets, $\{C_{i,j}\}_{j=1}^{\alpha_i}$, where each set, except possibly for the last one, contains $10M^2$ vertices, and if $v \in C_{i,j}$ and $u \in C_{i,j+1}$ then $\tau_i(v) \leq \tau_i(u)$. This completes the definition of the partition $C_{i,j}^h$ of $A_{h'}^{h'}$ induced by F_h. We denote the corresponding values α_i, for $1 \leq i \leq \log N$ by $\alpha_i^{h',h}$, omitting the superscripts when clear from context. We now define $R_{h',h} = \bigcup_{i=1}^{\log N} C_{i,1}^h$. Observe that $|R_{h',h}| \leq O(\log N \cdot M^2) = O(\log^3 N)$. We set $R_{h'} = \bigcup_{h=h'+1}^{M} R_{h',h}$, and so $|R_{h'}| \leq O(\log^4 N)$. This completes the first part of the proof.

From now on we fix an index $h : 1 \leq h \leq M$, and assume that $h \in I_q$ for some $1 \leq q \leq M$. Our goal is to prove that event $\mathcal{E}_2(h)$ happens with probability at least $1 - 1/N^2$ w.r.t. the choice of sets $R_{h'}$ described

above. Consider some iteration $h' : 1 \leq h' < h$ of the algorithm, and the partition $\{C_{i,j}\}_{i,j}$ of $A_{h'}^{h'}$ induced by F_h. Fix some set $C_{i,j}$ of the partition, for some $1 \leq i \leq \log N$, $1 \leq j < \alpha_i$. Since each vertex in $C_{i,j}$ is added to $U \cup R_{h'}$ with probability at least $\frac{16 \cdot 2^{q'}}{M}$ in iteration h' and $|C_{i,j}| = 10M^2$, we expect at least $160M \cdot 2^{q'}$ vertices of $C_{i,j}$ to be added to $U \cup R_{h'}$ in iteration h'. Let $\mathcal{E}(h,h',i,j)$ be the good event that at least $80M \cdot 2^{q'}$ vertices of $C_{i,j}$ are added to $U \cup R_{h'}$. We now proceed as follows. First, we show that with high probability, events $\mathcal{E}(h,h',i,j)$ happen for all relevant indices h', i, j. This is done in the next claim, whose proof uses standard Chernoff bound together with the union bound. Next we show a weight transfer procedure, which ensures that for each $1 \leq h' < h$, if $\mathcal{E}(h,h',i,j)$ happens for all i and j, then each remaining active vertex in $A_h^{h'}$ receives a sufficient contribution to its LP-weight.

Claim 5.1. *Events $\mathcal{E}(h,h',i,j)$ hold for all $1 \leq h' < h$, $1 \leq i \leq \log N$, $1 \leq j < \alpha_i^{h',h}$ with probability at least $1 - 1/N^2$.*

Proof. Since $\alpha_i^{h',h} \leq N$ for all i, h, h', the number of possible choices of indices h', i, j is bounded by $N \cdot M \cdot \log N \leq O(N \log^2 N)$.

Consider a specific event $\mathcal{E}(h,h',i,j)$, and let $C_{i,j} \subseteq A_{h'}^{h'}$ be the corresponding set of vertices. Then $|C_{i,j}| = 10M^2$, and each vertex $v \in C_{i,j}$ is chosen to be in $U \cup R_{h'}$ with probability at least $16 \cdot 2^{q'}/M$, where $h' \in I_{q'}$. The expected number of vertices of $C_{i,j}$ added to U is at least $160M \cdot 2^{q'}$, and so from Chernoff bound, the probability that less than $80M \cdot 2^{q'}$ vertices are added is bounded by $e^{-10M} \leq 1/N^{10}$. So the probability that a specific event $\mathcal{E}(h,h',i,j)$ does not happen is at most N^{-10}, and using the union bound, the probability that any of the events $\mathcal{E}(h,h',i,j)$ does not happen is at most $1/N^2$. □

Weight Transfer Procedure We now focus on the partition $\{C_{i,j}\}_{i,j}$ induced by F_h on $A_{h'}^{h'}$ and define a weight transfer procedure for vertices of $A_h^{h'}$. For each such $i : 1 \leq i \leq \log N$, for each $j : 1 \leq j \leq \alpha_i$, let $S_{i,j} = \bigcup_{v \in C_{i,j}} D_i(v)$, and $S'_{i,j} = \bigcup_{v \in C_{i,j}} D_{i+1}(v)$.

Consider a pair of vertices $v \in C_{i,j}$, $u \in C_{i,j+1}$. By the definition of the partition $\{C_{i,j}\}$, $\tau_i(v) \leq \tau_i(u)$. So if $v' \in D_i(v)$, then it appears at layer $\mathbf{L}(v') \leq \tau_i(v)$. On the other hand, if $u' \in D_{i+1}(u)$, then it belongs to layer $\mathbf{L}(u') \geq \tau_i(u) \geq \tau_i(v) \geq \mathbf{L}(v')$ (due to the definition of sets $D_{i'}(u)$). So we can transfer LP-weight from vertices of $S_{i,j}$ to vertices of $S'_{i,j+1}$ without increasing the amortized cost $\overline{Y}_{\tau,h}$ for any value τ.

We have two types of weight transfer rules. First, for each $i : 1 \leq i \leq \log N$, for each $j : 1 \leq j < \alpha_i$, we transfer the LP-weight between $S_{i,j}$ and $S'_{i,j+1}$, as follows: if any vertex $v \in S_{i,j}$ becomes inactive, then half of its current LP-weight $y_h^{h'}(v)$ is evenly split among all vertices of $S'_{i,j+1}$. For each $1 \leq i \leq \log N$, for each $1 \leq j < \alpha_i$, we say that vertices of $S'_{i,j+1}$, are *covered* by the first weight transfer rule.

The second weight transfer rule only applies to sets $C_{1,j}$, where $1 \leq j \leq \alpha_1$. Observe that for each $v \in C_1$, $D_1(v)$ only consists of a single vertex. We are therefore guaranteed that for each $1 \leq j < \alpha_1$, if $v \in S_{1,j}$ and $v' \in S_{1,j+1}$ then $\mathbf{L}(v) \leq \mathbf{L}(v')$. The second weight transfer rule is that for every $1 \leq j < \alpha_1$, if any vertex $v \in S_{1,j}$ becomes inactive, then half of its current LP-weight $y_h^{h'}(v)$ is evenly split among all vertices of $S_{1,j+1}$. For each $1 \leq j < \alpha_1$, we say that every vertex in $S_{1,j+1}$ is covered by the second weight transfer rule. This finishes the definition of the weight transfer procedure. Our claim is that every vertex $u \in A_h^{h'}$ is either covered by one of the weight transfer rules, or its ancestor belongs to $R_{h',h}$. In the former case, we will show that u receives a sufficient contribution to its LP-weight, while in the latter case, u becomes inactive after iteration h'.

Claim 5.2. *If $u \in A_h^{h'}$ is not covered by either weight transfer rule, then its ancestor belongs to $R_{h',h}$.*

Proof. Consider some $u \in A_h^{h'}$, and assume that $u \in D_i(v)$ for some $v \in A_{h'}^{h'}$. We consider the following two cases.

First, if $i > 1$, then $v \in C_{i-1}$ because $|D_{i-1}(v)| = 2^{i-2}$ must hold. Assume that $v \in C_{i-1,j}$, for some $1 \leq j \leq \alpha_{i-1}$. If $j > 1$, then $v \in S'_{i-1,j}$, and it is therefore covered by the first weight transfer rule. If $j = 1$, then $v \in R_{h',h}$.

The second case is when $i = 1$. In this case, either $v \in C_{1,1}$, and then $v \in R_{h',h}$, or else $v \in C_{1,j}$ for some $j > 1$, and so vertex u is covered by the second weight transfer rule. □

It now remains to show that if events $\mathcal{E}(h,h',i,j)$ hold, then each vertex $u \in A_h^{h'}$ receives a sufficient contribution. This is done in the following claim.

Claim 5.3. *Let $h' \in I_{q'}$, and assume that at the beginning of iteration h' the LP-weight $y_h^{h'}(u) \geq 2^{q'-1}/M$ for all $u \in A_h^{h'}$. Assume also that events $\mathcal{E}(h,h',i,j)$ hold for all $1 \leq i \leq \log N$, $1 \leq j < \alpha_i$. Then the LP-weight of every vertex $u \in A_h^{h'+1}$ increases by at least $(2^{q'}/M)^2$ in iteration h', i.e. $y_h^{h'+1}(u) \geq y_h^{h'}(u) + (2^{q'}/M)^2$.*

Proof. Consider some vertex $u \in A_h^{h'+1}$. Since it remains active at the end of iteration h', its ancestor

in $F_{h'}$ was not added to $R_{h'}$, so u is covered by one of the weight transfer rules. Assume first that u is covered by the first weight transfer rule, so $u \in S'_{i,j+1}$, for some $1 < i \leq \log N$, $1 \leq j < \alpha_i$. Since event $\mathcal{E}(h,h',i,j)$ holds, at least $80M \cdot 2^{q'}$ vertices of $C_{i,j}$ have been added to $U \cup R_{h'}$ during iteration h'. Each such vertex has 2^{i-1} descendants in $S_{i,j}$, and each such descendant has LP-weight of at least $2^{q'-1}/M$. So overall, we have at least $\frac{1}{2} \cdot 2^{i-1} \cdot 80M \cdot 2^{q'} \cdot \frac{2^{q'-1}}{M} = 2^{i-1} \cdot 20 \cdot 2^{2q'}$ weight that is evenly split among the vertices of $S'_{i,j+1}$. Recall that $|C_{i,j+1}| \leq 10M^2$, and for each vertex $v \in C_{i,j+1}$, $|D_{i+1}(v)| \leq 2^i$. Therefore, $|S'_{i,j+1}| \leq 10M^2 \cdot 2^i$. Each vertex $u \in S'_{i,j+1}$ then receives at least $\frac{20 \cdot 2^{i-1} \cdot 2^{2q'}}{10M^2 \cdot 2^i} \geq \frac{2^{2q'}}{M^2}$ weight as desired.

Assume now that $u \in S_{1,j+1}$. Again, since event $\mathcal{E}(h,h',1,j)$ happened, at least $80M \cdot 2^{q'}$ vertices of $C_{1,j}$ have been added to $U \cup R_{h'}$ in iteration h'. We then have at least $80M \cdot 2^{q'}$ vertices in $S_{1,j}$ that become inactive, each of which having LP-weight of $2^{q'-1}/M$. So each vertex of $S_{1,j+1}$ receives a contribution of at least $\frac{\frac{1}{2} \cdot 80 \cdot 2^{q'} \cdot 2^{q'-1}}{10M^2} \geq 2^{2q'}/M^2$. □

COROLLARY 5.1. *Let $h \in I_q$. Assume that events $\mathcal{E}(h,h',i,j)$ happen for all $1 \leq h' < h$, $1 \leq i \leq \log N$, $1 \leq j < \alpha_i$. Then for each phase $q' : 1 \leq q' \leq q$, for each vertex $u \in A_h^{(q')}$ that is active at the beginning of phase q', its LP-weight $y_h^{(q')}(u) \geq 2^{q'-1}/M$.*

Proof. The proof is by induction on q'. At the beginning of the first phase, the LP-weight of every vertex $u \in A_h^{(1)}$ for all h is $1/M$, so the claim holds. Assume now that it holds at the beginning of phase q'. Let $u \in A_h^{(q'+1)}$ be some vertex that remains active at the end of phase q'. We apply Claim 5.3 for all $h' \in I_{q'}$. This ensures that, for each one of the $M/2^{q'}$ iterations of phase q', the LP-weight of u increases by at least $(2^{q'}/M)^2$, and so the total increase in the LP-weight in phase q' is at least $2^{q'}/M$. □

We are now ready to complete the proof of Theorem 4.2. From the above corollary, if events $\mathcal{E}(h,h',i,j)$ happen for all $1 \leq h' < h$, $1 \leq i \leq \log N$, $1 \leq j < \alpha_i$, then at the beginning of iteration h, the LP-weight of every vertex in A_h^h is at least $2^{q-1}/M$. Moreover, we have only transferred weight from vertices in $\mathcal{L}_{\tau,h}$ to vertices in $\mathcal{L}_{\tau',h}$ for $\tau \leq \tau'$. Therefore, throughout the algorithm, the amortized LP-weight $\overline{Y}_{\tau,h}$ does not increase for any τ. So at the beginning of iteration h, the total LP-weight of active vertices in sets $\mathcal{L}_{1,h}, \ldots, \mathcal{L}_{\tau,h}$ is at least $\sum_{\tau' \leq \tau} |A_{\tau',h}| \cdot 2^{q-1}/M$ and at most $\overline{Y}_{\tau,h}$. Therefore, event $\mathcal{E}_2(h)$ holds with probability at least $1 - 1/N^2$. We have also shown that the sizes of the selected sets $R_{h'}$ are bounded by $O(\log^4 N)$.

6 Integrality Gap

In this section we present a lower bound of $\Omega(\log^* n)$ on the integrality gap of (LP), matching the upper bound to within a constant factor. There is a natural way to strengthen the LP as follows. Guess the value k of the optimal solution and scan the non-terminal vertices in the reversed order of their layers: $L_\lambda, L_{\lambda-1}, \ldots, L_1$. When vertex v is considered, apply (LP) to instance (G_v, T_v), where G_v is the sub-tree rooted at v and T_v is the subset of terminals contained in it. If the cost of the LP solution for (G_v, T_v) is greater than k, then v cannot lie on a fire spreading path in the optimal solution and is therefore protected by it. We then remove from G all vertices of G_v except for v, and set the new set of terminals to be $(T \backslash T_v) \cup \{v\}$. Let G' be the tree obtained after all vertices of G have been processed. Then either G' consists of a single vertex s, with the set of terminals $T = \{s\}$, so we obtain a certificate that the guessed value k of the optimal solution is too low, or for each vertex v of G', there is a feasible solution of cost at most k for (LP) on the instance (G'_v, T_v) defined by the sub-tree of G' rooted at v. In the latter case we say that G' is k-feasible. We show that the integrality gap of (LP) is $\Omega(\log^* n)$ even on 1-feasible instances.

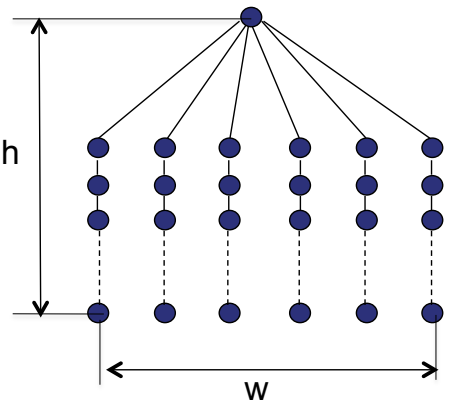

Figure 1: Type-1 spider of width w and height h

6.1 Integrality Gap Construction We use the following two simple spider graphs. A type-1 spider of width w and height h consists of w paths of length h each. The paths are completely disjoint, except that they all share one common endpoint called the *spider head*. The other endpoints of the w paths are called the *spider feet*. Type-2 spider of width w and height h is

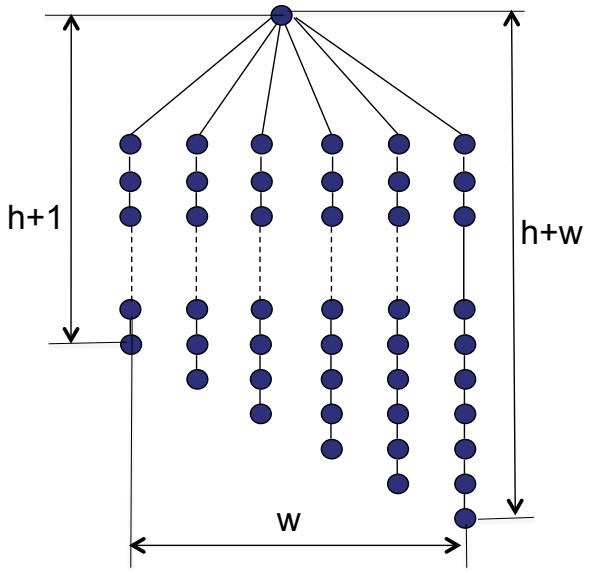

Figure 2: Type-2 spider of width w and height h

defined similarly, except that the lengths of the w paths vary, with the ith path, for $1 \leq i \leq w$, having length $h+i$ (see Figures 1 and 2). We view each such spider as a tree rooted at its head.

The integrality gap instance G uses a parameter M, and it consists of $M+1$ *levels*. Each level $i : 0 \leq i \leq M$ is a forest. The set of roots of its trees is denoted by $\mathcal{A}_i$, and the set of their leaves by $\mathcal{B}_i$. All vertices of $\mathcal{A}_i$ lie in the same layer of G, and the same is true for $\mathcal{B}_i$. For convenience, when defining level i, we renumber its layers, so $\mathcal{A}_i$ is layer 0 of level i and $\mathcal{B}_i$ is the last layer, whose index is denoted by λ_i. Vertices in set $\mathcal{B}_i$ are partitioned into ℓ_i subsets, $\mathcal{B}_i = \bigcup_{j=1}^{\ell_i} \mathcal{B}_i^j$. We now proceed to describe the levels. Level 0 consists of only two layers, $\mathcal{A}_0 = \{s\}$ and $\mathcal{B}_0 = \{s_1, \ldots, s_{2M}\}$ with edges connecting s to every vertex of $\mathcal{B}_0$. Set $\mathcal{B}_0$ is partitioned into $\ell_0 = 2M$ subsets $\mathcal{B}_0^1, \ldots, \mathcal{B}_0^{2M}$, with $\mathcal{B}_0^j = \{s_j\}$ for $1 \leq j \leq 2M$. We now describe level i for some $i > 0$. Consider the last layer $\mathcal{B}_{i-1}$ of level $i-1$ and its corresponding partition $\left\{\mathcal{B}_{i-1}^j\right\}_{j=1}^{\ell_{i-1}}$. Let I be the set of all ordered M-tuples of vertices in $\mathcal{B}_{i-1}$, where the vertices in each M-tuple belong to distinct sets of the partition. Therefore, I consists of all ordered M-tuples $(v_1, \ldots, v_M)$, such that, for each $1 \leq r \leq M$, if $v_r \in \mathcal{B}_{i-1}^{j_r}$, then all indices $j_1, \ldots, j_M$ are distinct. Let $\ell_i = |I| \leq |\mathcal{B}_{i-1}|^M$. We add a set $\mathcal{A}_i^\rho = \{v_1^\rho, \ldots, v_M^\rho\}$ of vertices for each $\rho \in I$ and set $\mathcal{A}_i = \bigcup_{\rho \in I} \mathcal{A}_i^\rho$. For each $\rho = (v_1, \ldots, v_M) \in I$, for each $r : 1 \leq r \leq M$, there is an edge (v_r, v_r^ρ), where $v_r \in \mathcal{B}_{i-1}$. This finishes the definition of set $\mathcal{A}_i$. Level i contains, for each M-tuple $\rho = (v_1^\rho, \ldots, v_M^\rho) \in I$, a gadget H_i^ρ, consisting of M identical trees, whose roots are the M vertices of $\mathcal{A}_i^\rho$. The set of the leaves of the M trees, lying in layer λ_i, is denoted by $\mathcal{B}_i^\rho$. We then set $\mathcal{B}_i = \bigcup_{\rho \in I} \mathcal{B}_i^\rho$, with the corresponding partition of $\mathcal{B}_i$ into ℓ_i subsets $\{\mathcal{B}_i^\rho\}_{\rho \in I}$. We set $\lambda_i = \sum_{j=1}^{\ell_i} (2M)^j$, and we assume that we are given some arbitrary ordering $\rho_1, \ldots, \rho_{\ell_i}$ of the tuples in I.

We now fix an M-tuple $\rho_j = (v_1, \ldots, v_M) \in I$ and define the corresponding gadget $H_i^{\rho_j}$. The gadget consists of M copies of tree T_j, rooted at vertices $v_1^{\rho_j}, \ldots, v_M^{\rho_j}$. Let v denote the root of T_j, and let $b_j = \sum_{1 \leq j' < j} (2M)^{j'}$. We add a type-2 spider of width $(2M)^j$ and height b_j, whose head is v. The feet of the spider are called *special vertices*. If the current level $i \neq M$, then we perform the following additional step. For each special vertex u of T_j, if u lies at layer $b_j + h$, for $1 \leq h \leq (2M)^j$, then we add a type-1 spider of width $2M\lambda_i$ and height $\lambda_i - h - b_j$ whose head is u. This ensures that all leaves of tree T_j lie in layer λ_i. The final gadget $H_i^{\rho_j}$ is obtained by attaching a copy of T_j as a subtree to each vertex in $\mathcal{A}_i^{\rho_j}$. This finishes the description of level i. Notice that there are at most M special vertices at each layer of level i. The set T of terminals is the set of all special vertices lying at the last level M. Let N denote the construction size.

6.2 Analysis We show that instance G is 1-feasible, and in particular the cost of the fractional solution is 1. On the other hand, we prove that the cost of any feasible integral solution is at least M. We start by bounding N, the construction size.

Since $N \leq O(|\mathcal{B}_M|^2)$, it is enough to bound the sizes of sets $\mathcal{B}_i$ for all i. Recall that $|\mathcal{B}_0| = 2M$. For $i \geq 1$, we have that $\ell_i \leq |\mathcal{B}_{i-1}|^M$. Therefore, $\lambda_i \leq (2M)^{\ell_i+1} \leq (2M)^{|\mathcal{B}_{i-1}|^M+1}$. The number of special vertices at level i is at most $M\lambda_i$, and each special vertex has $2M\lambda_i$ descendants in set $\mathcal{B}_i$, so overall $|\mathcal{B}_i| \leq 2M^2\lambda_i^2 \leq 2M^2 \cdot (2M)^{2|\mathcal{B}_{i-1}|^M+2} \leq (2M)^{|\mathcal{B}_{i-1}|^{2M}}$. For simplicity, denote $Y_i = |\mathcal{B}_i|$ for all i, and $m = 2M$. We then have the following recurrence: $Y_0 = m$, and for $0 \leq i < M$, $Y_{i+1} \leq m^{Y_i^m}$. We use the following claim.

Claim 6.1. *For $0 \leq i \leq M$, $\log^{(i)} N \leq m^2 Y_{M-i}^m$.*

Proof. The claim holds trivially for $i = 0$. Assume that the claim holds for some i. Then $\log^{(i+1)} N \leq \log(m^2 Y_{M-i}^m) \leq 2\log m + m \log Y_{M-i}$. Replacing $Y_{M-i} \leq m^{Y_{M-i-1}^m}$, we get that:

$$\begin{aligned}\log^{(i+1)} N &\leq 2\log m + m\log\left(m^{Y^m_{M-i-1}}\right)\\ &\leq 2\log m + m\cdot Y^m_{M-i-1}\log m \leq m^2 Y^m_{M-i-1}\end{aligned}$$

for a large enough m. □

Applying the claim for $i = M$, we obtain that $\log^{(M)} N \leq m^2 Y^m_0 \leq (2M)^{O(M)}$. Clearly, taking the logarithm for $O(\log^* M) \leq M$ more steps will get the number on the right hand side below 1. Therefore, $\log^{(2M)} N \leq 1$, and $M = \Omega(\log^* N)$.

We now to proceed to analyze the fractional solution cost.

Claim 6.2. *There is a fractional solution of cost* 1 *for instance G, and moreover G is* 1*-feasible.*

Proof. The fractional solution simply assigns a $1/M$ value to each special vertex. Since every layer contains at most M special vertices, the cost of the solution is 1. It is also easy to see that for every level i, if $u \in \mathcal{B}_i$ is a descendant of $v \in \mathcal{A}_i$, then the path connecting v to u contains exactly one special vertex. Therefore, there are M special vertices on every root-to-terminal path, and the solution is feasible.

We now show that instance G is 1-feasible. Consider some non-terminal vertex v, its subtree G_v and the corresponding subset T_v of terminals. If v has only one child, then there is a trivial fractional solution of cost 1 to instance (G_v, T_v), in which an LP-value of 1 is assigned to the child of v. Assume now that v has more than 1 child. This can only happen if v is a special vertex, or $v \in \mathcal{A}_i$, or $v \in \mathcal{B}_i$ for some level i.

Assume first that $v \in \mathcal{A}_i$ for some level $i : 1 \leq i \leq M$ (the case where $i = 0$, and $v = s$ has been analyzed above). Consider the sub-tree G^i_v rooted at v, restricted to only vertices of level i. This subtree contains at most one special vertex at each layer. Therefore, there is a feasible solution of cost 1 to instance (G_v, T_v), where LP-weight of 1 is assigned to every special vertex of G^i_v.

Assume now that $v \in \mathcal{B}_{i-1}$ for some $i : 1 \leq i \leq M$. Let $v^{\rho_1}, \ldots, v^{\rho_h}$ be the children of v. Due to the definition of I, each vertex $v^{\rho_j} : 1 \leq j \leq h$ participates in a distinct gadget H^ρ_i at level i. This again ensures that all special vertices lying in the sub-tree of v at level i belong to distinct layers. A feasible solution of cost 1 is then obtained by placing an LP-weight of 1 on each such special vertex.

Finally, assume that v is a non-terminal special vertex at level $i : 1 \leq i < M$. Due to the definition of the partition of $\mathcal{B}_i$, all descendants of v in $\mathcal{B}_i$ belong to the same set $\mathcal{B}^j_i$ of the partition. This ensures that all descendants of v in $\mathcal{A}_{i+1}$ participate in distinct gadgets H^ρ_{i+1}, and therefore the special vertices in the sub-tree of v lying at level $i+1$ all belong to distinct layers. Again, a feasible fractional solution of cost 1 for the corresponding instance is obtained by assigning an LP-value 1 to all special vertices lying at level $i+1$. □

It now only remains to show that the optimal integral solution cost is at least M. Let λ be the total number of layers in our construction, and let $\mathcal{S} = \{U_\tau\}_{\tau \geq 1}$ be any solution of cost at most $M-1$. From Observation 2.1, we can assume, w.l.o.g., that for all $1 \leq \tau \leq \lambda$, $U_\tau \subseteq L_\tau$, where L_τ is layer τ of the tree, and $U_\tau = \emptyset$ for $\tau > \lambda$. We assume that we are given a solution $\mathcal{S} = \{U_\tau\}^\lambda_{\tau=1}$ of this form. To simplify the analysis of this part, we view the set of vertices $\mathcal{B}_0 = \{s_1, \ldots, s_{2M}\}$ as special vertices, and we define $2M$ gadgets $H^\rho_0 : 1 \leq \rho \leq 2M$ at level 0, where gadget H^ρ_0 consists of a single vertex s_ρ. In order to prove that $\mathcal{S}$ is not a feasible solution, it is enough to show that at least one terminal is not protected. The next lemma will then complete the analysis of the integrality gap.

Lemma 6.1. *Let $\mathcal{S} = \{U_\tau\}^\lambda_{\tau=1}$ be any integral solution of cost at most $M-1$. Then for every $i : 0 \leq i \leq M$, there is a set V_i of $M+1$ special vertices lying at level i, that are not protected by $\mathcal{S}$. Moreover, all vertices in V_i belong to distinct level-i gadgets H^ρ_i.*

Proof. The proof is by induction. Consider first $i = 0$. Since $\mathcal{B}_0$ contains $2M$ vertices, and the solution is only allowed to save $M-1$ of them, at least $M+1$ vertices in $\mathcal{B}_0$ are not protected, and they belong to distinct gadgets by definition. Assume now that the lemma is true for some $0 \leq i < M$, and let $V_i = \{u_1, \ldots, u_{M+1}\}$ be the corresponding set of unprotected special vertices at level i. We need the following claim:

Claim 6.3. *Let u be any special vertex lying at level i that is not protected by $\mathcal{S}$. Then at least one descendant of u in $\mathcal{B}_i$ is not protected by $\mathcal{S}$.*

Proof. Assume otherwise. Recall that there is a type-1 spider of width $2M\lambda_i$ and height at most λ_i rooted at u. Let Q be the set of the leaves of this spider, $Q \subseteq \mathcal{B}_i$. Protecting each vertex in Q requires saving a distinct vertex of the spider. As vertex u is not protected, solution $\mathcal{S}$ can only protect at most $(M-1)\lambda_i$ vertices of the spider, and hence protect at most $(M-1)\lambda_i$ vertices of Q. □

For each vertex $u_j \in V_i$, let w_j be any descendant of u_j in $\mathcal{B}_i$ that is not protected by $\mathcal{S}$. Since vertices $\{u_j\}^{M+1}_{j=1}$ belong to distinct level-i gadgets, vertices $w_1, \ldots, w_{M+1}$ belong to distinct sets in the partition of $\mathcal{B}_i$. We say that an M-tuple $\rho \in I$ is bad iff

$\rho \subseteq \{w_1, \ldots w_{M+1}\}$. Clearly, there are at least $2M$ bad ordered tuples $\rho \in I$. Given a bad tuple $\rho \in I$, we say that the corresponding level-$(i+1)$ gadget H^{ρ}_{i+1} is bad iff none of the vertices in set $\mathcal{A}^{\rho}_{i+1}$ is protected by $\mathcal{S}$. Since we have at least $2M$ bad tuples ρ, there are $2M$ corresponding disjoint sets $\mathcal{A}^{\rho}_{i+1}$ of vertices in $\mathcal{A}_{i+1}$, while $\mathcal{S}$ can only save $M-1$ vertices of $\mathcal{A}_{i+1}$. Therefore, there are at least $M+1$ bad level-$(i+1)$ gadgets. It now only remains to prove that each such bad gadget contains at least one special vertex that is not protected by $\mathcal{S}$.

Claim 6.4. *Let $H = H^{\rho}_{i+1}$ be a bad level-$(i+1)$ gadget. Then at least one special vertex of H is not protected by $\mathcal{S}$.*

Proof. Assume that ρ is the jth tuple in the corresponding set I. Consider the set $\mathcal{A}^{\rho}_{i+1}$ of vertices. Each vertex $v \in \mathcal{A}^{\rho}_{i+1}$ is a head of a type-2 spider of width $(2M)^j$ and height b_j, and none of the vertices in $\mathcal{A}^{\rho}_{i+1}$ is protected by $\mathcal{S}$. Therefore, in order to protect all special vertices of these spiders, $\mathcal{S}$ has to save at least $M \cdot (2M)^j$ vertices in layers $1, \ldots, b_j + (2M)^j$ of level $(i+1)$. Since at most $(M-1)$ vertices can be saved in each layer, it is enough to show that $M \cdot (2M)^j > (M-1)(b_j + (2M)^j)$, or equivalently that $(2M)^j > (M-1)b_j$. Indeed, $b_j = \sum_{j'=1}^{j-1} (2M)^{j'} \leq \frac{(2M)^j}{2M-1} < \frac{(2M)^j}{M-1}$. □

□

7 RMFC on Directed Layered Graphs

In this section we show a simple $O(\log n)$-approximation LP-rounding algorithm for RMFC on directed layered graphs, matching the lower bound of [12] on the integrality gap of the LP. For completeness, we sketch their lower bound in Appendix. We assume that we are given a directed graph $G = (V, E)$, whose vertices are partitioned into layers $L_0, \ldots, L_\lambda$, where $L_0 = \{s\}$, and all edges are between consecutive pairs of layers $L_\tau, L_{\tau+1}$, directed from L_τ towards $L_{\tau+1}$. Let OPT denote the optimal solution cost. Notice that given any solution $\mathcal{S} = \{U_\tau\}_{\tau=1}^{n}$, if vertex $v \in L_{\tau'}$ is not protected by $\mathcal{S}$, then it starts burning at time τ', so we can assume that $v \notin U_\tau$ for $\tau > \tau'$. We therefore have the following observation:

Observation 7.1. *Given any feasible solution $\{U_\tau\}_{\tau=1}^{n}$ for RMFC on directed layered graphs, we can assume w.l.o.g. that for every $\tau : 1 \leq \tau \leq \lambda$, $U_\tau \subseteq \bigcup_{\tau' \geq \tau} L_{\tau'}$, and $U_\tau = \emptyset$ for all $\tau > \lambda$.*

For vertex $v \in V$, let $\mathcal{P}_v$ be the set of all paths connecting the source s to v. Similarly to RMFC on trees, we define amortized solution cost. Let $U \subseteq V$ be any subset of vertices. We say that U is a *feasible set* iff for each terminal $t \in T$, for every path $P \in \mathcal{P}_t$, $P \cap U \neq \emptyset$. The *amortized cost* of U is the maximum, over all $1 \leq \tau \leq \lambda$, of $\left(\sum_{\tau' \leq \tau} |U \cap L_{\tau'}|\right)/\tau$. Let OPT' denote the minimum amortized cost of any feasible set U. Similarly to RMFC on trees, the next claim shows that $\mathsf{OPT} = \mathsf{OPT}'$.

Claim 7.1. *For any RMFC instance on directed layered graphs, $\mathsf{OPT} = \mathsf{OPT}'$.*

Proof. Let $\{U_\tau\}_{\tau=1}^{\lambda}$ be the optimal solution to the RMFC instance, and let $U = \bigcup_{\tau=1}^{\lambda} U_\tau$. From Observation 7.1, $\sum_{\tau' \leq \tau} |U \cap L_{\tau'}| \leq \sum_{\tau' \leq \tau} |U_{\tau'}| \leq \tau \mathsf{OPT}$ for all τ, so the amortized cost of U is at most OPT.

We now show the opposite direction. Let U be a feasible set with amortized cost k. We construct a solution $\{U'_\tau\}_{\tau=1}^{\lambda}$ with (non-amortized) cost at most k as follows. Consider first the solution $\mathcal{S} = \{U_\tau\}_{\tau=1}^{\lambda}$, where $U_\tau = U \cap L_\tau$ for all τ. It is easy to see that $\mathcal{S}$ is a feasible solution: assume otherwise, and let t be any terminal that is not protected by $\mathcal{S}$. Let $P = (s, v_1, \ldots, v_z = t)$ be the corresponding fire spreading path. Since U is a feasible set, there is some $v_\tau \in P \cap U$, for $1 \leq \tau \leq z$. However, $v_\tau \in U_\tau$, and so P cannot be a fire spreading path.

Finally, we transform the solution $\mathcal{S}$ to ensure that its (non-amortized) cost is at most k. This is done similarly to the proof of Claim 2.1, as follows. We scan the sets $U_\lambda, U_{\lambda-1}, \ldots, U_1$ in this order. Let U_τ be the current set. If $|U_\tau| = k' > k$, then we remove any collection of $k' - k$ vertices from U_τ and add them to $U_{\tau-1}$. It is easy to see that this operation does not increase the amortized cost and preserves the feasibility of the solution. Once all the sets U_τ are processed, each set contains at most k vertices. □

We can therefore focus on finding a feasible set with a low amortized cost. We use the following LP relaxation.

$$\begin{array}{lll} & \min & k \\ \text{s.t.} & \sum_{\tau' \leq \tau} \sum_{v \in L_{\tau'}} x(v) \leq \tau k & \forall \tau : 1 \leq \tau \leq \lambda \\ & \sum_{v \in P} x(v) \geq 1 & \forall t \in T, \forall P \in \mathcal{P}_t \\ & x(v) \geq 0 & \forall v \in V \end{array}$$

Though the number of constraints in this LP is exponential, it can be efficiently solved using a separation oracle, that computes a minimum-weight path connecting s to any terminal $t \in T$. We now describe

our LP-rounding algorithm. First, we round each LP-value $x(v)$ up to the next multiple of $1/n$. The resulting fractional solution is clearly still feasible, and its amortized cost increases by at most a factor of 2. Next, we partition the vertices with non-zero LP-weight into n sets $F_1, \ldots, F_n$, as follows. Let $v \in V$ be any vertex with $x(v) = r/n$, where $r > 0$. For each path $P \in \mathcal{P}_v$, let $X(P) = \sum_{v' \in P \setminus \{v\}} x(v')$, and let $h = \min_{P \in \mathcal{P}_v} \{n \cdot X(P)\}$. Then v belongs to sets $F_{h+r'}$ for all $1 \leq r' \leq r$. We view the weight $x(v)$ of v as being evenly distributed among these r sets, so the weight of v w.r.t. $F_{h+r'}$ is $x_{h+r'}(v) = 1/n$, for $1 \leq r' \leq r$. We need the following claim.

CLAIM 7.2. *For each $h : 1 \leq h \leq n$, set F_h is a feasible set.*

Proof. Let $t \in T$ be any terminal and let $P = (s = v_0, v_1, \ldots, v_z = t) \in \mathcal{P}_t$ be any path connecting s to t. It is enough to show that for each $h : 1 \leq h \leq n$, $P \cap F_h \neq \emptyset$. For each $\tau : 0 \leq \tau \leq z$, let $Y_\tau = \min_{P \in \mathcal{P}_{v_\tau}} \{X(P)\}$. Let τ' be the largest index for which $Y_{\tau'} < h/n$. We argue that $v_{\tau'} \in F_h$. First notice that $v_{\tau'}$ belongs to $F_{h'+1}, \ldots, F_{h'+r}$ for $h' = nY_{\tau'} < h$ and $r = nx(v_{\tau'})$. There are two cases. If $\tau' = z$, we have that $Y_{\tau'} + x(v_{\tau'}) \geq 1$, and therefore $v_{\tau'} \in F_h$ since $h' < h \leq h' + r$. Otherwise if $\tau' < z$, we have $Y_{\tau'} + x(v_{\tau'}) \geq Y_{\tau'+1} \geq h/n$, and hence $v_{\tau'} \in F_h$ since, again, $h' < h \leq h' + r$. □

Our final solution is the set F_{h^*} with smallest amortized cost over all sets F_h. The next claim shows that this gives an $O(\log n)$-approximation.

CLAIM 7.3. *The amortized cost of F_{h^*} is at most $O(k \log n)$.*

Proof. We first recall the notation used in the previous sections. We denote, for each $h : 1 \leq h \leq n$ and each $\tau : 1 \leq \tau \leq \lambda$, $X_{\tau,h} = \sum_{v \in F_h \cap L_\tau} x_h(v)$ and $\overline{X}_{\tau,h} = \sum_{\tau' \leq \tau} X_{\tau',h}$. For each h, let τ_h denote the layer $\tau : 1 \leq \tau \leq \lambda$ that maximizes $\overline{X}_{\tau,h}/\tau$.

For each h, we denote the amortized cost of F_h by by $\text{COST}(F_h) = \max_{\tau \leq \lambda} \left\{ \frac{1}{\tau} \left(\sum_{\tau' \leq \tau} |F_h \cap L_{\tau'}| \right) \right\}$, or equivalently, $\text{COST}(F_h) = \max_\tau \left\{ \frac{1}{\tau} \sum_{\tau' \leq \tau} X_{\tau',h} \cdot n \right\} = \frac{n}{\tau_h} \cdot \overline{X}_{\tau_h,h}$, since the LP-weight $x_h(v)$ of every vertex $v \in F_h$ w.r.t. F_h is $1/n$. We now have that:

$$\text{COST}(F_{h^*}) \leq \frac{1}{n} \sum_{h=1}^{n} \text{COST}(F_h) = \sum_{h=1}^{n} \overline{X}_{\tau_h,h}/\tau_h$$

Partition the indices $h : 1 \leq h \leq n$ into $\lceil \log \lambda \rceil$ subsets $I_1, \ldots, I_{\lceil \log \lambda \rceil}$, as follows: $I_q = \left\{ h \mid 2^{q-1} \leq \tau_h < 2^q \right\}$, for $1 \leq q \leq \lceil \log \lambda \rceil$. For each such set I_q, we bound the summation for indices $h \in I_q$ as follows:

$$\begin{aligned} \sum_{h \in I_q} \overline{X}_{\tau_h,h}/\tau_h &\leq \sum_{h \in I_q} \overline{X}_{2^q,h}/2^{q-1} \\ &\leq 2 \sum_{h=1}^{n} \overline{X}_{2^q,h}/2^q \\ &\leq 2\overline{X}_{2^q}/2^q \leq 4k \end{aligned}$$

(the additional factor of 2 comes from the original rounding to produce a $1/n$-integral solution). Summing up over all sets I_q, we get that $\text{COST}(F_{h^*}) \leq O(\mathsf{OPT} \cdot \log \lambda) \leq O(\log n)\mathsf{OPT}$. □

Acknowledgement: We would like to thank Sanjeev Khanna and Neil Olver for suggesting the problem to us, and for many helpful discussions.

References

[1] E. Anshelevich, D. Chakrabarty, A. Hate and C. Swami. Approximations for the FireFighter Problem: Cuts over Time and Submodularity. ISAAC 2009, to appear.

[2] A. F. Archer. Two $O(\log^* k)$-approximation algorithms for the asymmetric k-center problem. *Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization*, pp. 1–14, 2001.

[3] L. Cai, E. Verbin, and L. Yang. Firefighting on Trees: $(1-1/e)$-Approximation, Fixed Parameter Tractability and a Subexponential Algorithm. *ISAAC* 2008

[4] J. Chuzhoy, S. Guha, E. Halperin, G. Kortsarz, S. Khanna, R. Krauthgamer and S. Naor. Asymmetric k-center is log*n-hard to Approximate. *Journal of the ACM*, Volume 52, Issue 4, pp. 538-551, 2005.

[5] M. Develin and S. Hartke. Fire containment in grids of dimension three and higher. *Discrete Applied Mathtematics*, 155(17), 2257-2268 (2007)

[6] S Finbow, A King, G MacGillivray, R Rizzi The firefighter problem for graphs of maximum degree three. *Discrete Mathematics*, 307(16), 2094-2105 (2007)

[7] P. Fogarty. Catching the fire on grids M.Sc. Thesis, University of Vermont, 2003.

[8] B.L. Hartnell. Firefighter! An application of domination. *24th Manitoba Conference on Combinatorial Mathematics and Computing*, 1995.

[9] S.G. Hartke. Attempting to narrow the integrality gap for the firefighter problem on trees. *Discrete Methods in Epidemiology, J.Abello and G.Cormode, eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 70 (2006), 179-185.

[10] S.G. Hartke. Graph-theoretic models of spread and competition. *Phd. Thesis*, Rutgers, The State University of New Jersey.

[11] B. Hartnell and Q. Li Firefighting on trees: How bad is the greedy algorithm? Congressus Numerantium, 2000
[12] S. Khanna, N. Olver. Personal communication, 2008.
[13] A. King and G. MacGillivray The firefighter problem for cubic graphs. Discrete Mathematics, to appear
[14] G. MacGillivray, P. Wang. On the firefighter problem *J. Comb. Math. Comb. Comp.*, 47, 83-96 (2003).
[15] M.E. Messinger. Firefighting on the Triangular Grid. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 63 (2007) 37-45.
[16] M.E. Messinger Firefighting on the Strong Grid. Manuscript.
[17] M.E. Messinger. Average Firefighting on Infinite Grids. *Australasian Journal of Combinatorics*, 41 (2008) 15-28.
[18] K.L.Ng and P. Raff. A generalization of the firefighter problem on $\mathbb{Z} \times \mathbb{Z}$. *Discrete Applied Mathematics*, 156(5), 730-745 (2008).
[19] R. Panigrahy and S. Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *J. of Algorithms*, 27(2):259–268, 1998.
[20] P. Wang and G. Moeller. Fire control on graphs *J. Comb. Math. Comb. Comp.*, 41, 19-34 (2002).

A Chernoff Bound

We state Chernoff bound that we use throughout the paper.

THEOREM A.1. (CHERNOFF BOUND) *Let $X_1, \ldots, X_n$ be independent Poisson random variables such that $\mathbf{Pr}[X_i = 1] = p_i$. Let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbf{E}[X]$. Then:*

- *For $0 < \delta < 1$, $\mathbf{Pr}[X \leq (1-\delta)\mu] \leq e^{-\mu\delta^2/2}$.*
- *For $R \geq 6\mu$, $\mathbf{Pr}[X \geq R] \leq 2^{-R}$*

B Proof of Claim 3.1

For $1 \leq i \leq g$, denote $Z_i = |T_{g-i+1}|$. We then have the following recursion: $Z_1 = |T_g| \leq n$, and for all $i > 1$, $Z_i \leq c \cdot \log^5 Z_{i-1}$, for some constant $c > 1$. We need to prove that Z_g is bounded by a constant for a suitably chosen $g = O(\log^* n)$.

We prove by induction that $Z_i \leq (10c \log^{(i-1)} n)^5$. The base case when $i = 1$ holds trivially. Assume that the statement holds for i and consider Z_{i+1}. Then $Z_{i+1} \leq c \log^5 Z_i \leq c\left(\log\left((10c\log^{(i-1)} n)^5\right)\right)^5$, by the induction hypothesis. We then get that $Z_{i+1} \leq c\left(5\log(10c) + 5\log^{(i)} n\right)^5 \leq \left(10c\log^{(i)} n\right)^5$ whenever $\log^{(i)} n > 5c\log(10c)$. So for $g = O(\log^* n)$, Z_g is bounded by a constant.

C Proof of Theorem 4.1

For each terminal $t \in \mathcal{T}$, let P_t denote the unique path connecting t to the root of H. We start by removing from H all vertices that do not lie on any path P_t for $t \in \mathcal{T}$. Let $H' = (V', E')$ be the resulting tree and $\mathcal{L}'_1, \ldots, \mathcal{L}'_\lambda$ be the resulting layers, so $\mathcal{L}'_\tau = V' \cap \mathcal{L}_\tau$ for all τ. Observe that since every vertex of V' lies on some path P_t for $t \in \mathcal{T}$, $|\mathcal{L}'_\tau| \leq N$ for all τ. Next, for each vertex $v \in V'$ whose distance from the root is at most N, we set $x'(v) = x(v) + 1/N$, and for all other vertices we set $x'(v) = 0$. Since H' is a tree with N leaves, the number of vertices with non-zero value $x'(v)$ is now at most N^2. It is easy to see that x' is a feasible fractional solution, as the summation of values $x'(v)$ along any path P_t for $t \in \mathcal{T}$ is at least 1: If P_t contains at least N vertices, then values $x'(v)$ of the first N vertices of P_t are at least $1/N$, and their total sum is at least 1. Otherwise, if P_t contains less than N vertices, then for each vertex $v \in P_t$, $x'(v) \geq x(v)$, so the total sum remains at least 1. For each $1 \leq \tau \leq \lambda$, let $X'_\tau = \sum_{v \in \mathcal{L}'_\tau} x'(v)$, and let $\overline{X}'_\tau = \sum_{\tau' \leq \tau} X'_{\tau'}$. Since $|\mathcal{L}'_\tau| \leq N$, we have that $X'_\tau \leq X_\tau + 1$ and $\overline{X}'_\tau \leq \tau + \overline{X}$ for all τ.

The next step is to perform randomized rounding. Let z be the largest index for which $\overline{X}'_z \leq \frac{1}{2}$. For vertices $v \in \mathcal{L}_\tau$ with $\tau \leq z$, we set $y(v) = 0$. Consider now any vertex $v \in \mathcal{L}'_\tau$ for $\tau > z$. We can write $x'(v) = \frac{c_v}{M} + p_v$ where $p_v < 1/M$ and $c_v = \lfloor x'(v)M \rfloor$. We set $y(v) = \frac{c_v + b_v}{M}$, where $b_v = 1$ with probability Mp_v, and it is 0 otherwise. We first show that the resulting solution y is "almost feasible" w.h.p.:

CLAIM C.1. *With probability at least $1 - 1/N^2$, for each terminal $t \in \mathcal{T}$, $\sum_{v \in P_t} y(v) \geq 1/8$.*

Proof. Fix some terminal $t \in T$. Let $P'_t \subseteq P_t$ contain all vertices lying in layers $\mathcal{L}'_\tau$ with $\tau > z$. Since the total weight $\overline{X}'_z$ of all vertices in layers $\mathcal{L}'_1, \ldots, \mathcal{L}'_z$ is bounded by $\frac{1}{2}$, $\sum_{v \in P'_t} x'(v) \geq \frac{1}{2}$. We now consider two cases. If $\sum_{v \in P'_t} c_v/M \geq 1/4$, then clearly $\sum_{v \in P_t} y(v) \geq 1/4$. Otherwise, $\sum_{v \in P'_t} p_v \geq 1/4$. Consider the random variables b_v for $v \in P'_t$. These are independent $\{0,1\}$ variables with $\mathbf{E}\left[\sum_{v \in P'_t} b_v\right] = M \sum_{v \in P'_t} p_v \geq M/4$. Using Chernoff bound:

$$\mathbf{Pr}\left[\sum_{v \in P'_t} y(v) \leq 1/8\right] \leq \mathbf{Pr}\left[\sum_{v \in P'_t} b_v \leq M/8\right] \leq e^{-M/32}$$

Using the union bound over all terminals, and the fact that $M \geq 100 \log N$, we get that with probability at least $1 - 1/N^2$, for each terminal $t \in \mathcal{T}$, $\sum_{v \in P_t} y(v) \geq 1/8$. □

We now bound the amortized cost of the solution y.

CLAIM C.2. *With probability at least* $1-1/N^2$, *for each* $\tau : 1 \le \tau \le \lambda$, $\overline{Y}_\tau \le 8\overline{X}'_\tau$.

Proof. The proof is again a simple application of the Chernoff bound. Let I be the set of indices τ for which $\mathcal{L}'_\tau$ contains at least one vertex v with non-zero value $x'(v)$. Recall that $|I| \le N^2$. It is enough to prove that the claim holds for all $\tau \in I$, since for $\tau \notin I$, $Y_\tau = 0$. For $\tau \le z$, $\overline{Y}_\tau = 0$ so the claim clearly holds.

Consider now some $\tau \in I$, $\tau > z$. Let S contain all vertices $v \in \mathcal{L}'_{\tau'}$ for all $\tau' : z < \tau' \le \tau$, so $\overline{Y}_\tau = \sum_{v \in S} y(v)$. Let $C_\tau = \sum_{v \in S} c_v$ and $P_\tau = \sum_{v \in S} p_v$, so $\overline{X}'_\tau \ge C_\tau/M + P_\tau$. Let $B_\tau = \sum_{v \in S} b_v$, so $\overline{Y} = (C_\tau + B_\tau)/M$. Recall that $\overline{X}'_\tau \ge \frac{1}{2}$ for $\tau > z$.

We now consider two cases. First, if $P_\tau \ge \frac{1}{2}$, then $\mathbf{E}[B_\tau] = MP_\tau \ge M/2$. Therefore, using Chernoff bound, $\mathbf{Pr}[B_\tau \ge 6MP_\tau] \le 2^{-3M}$. But if $B_\tau \le 6MP_\tau$, then $\overline{Y}_\tau = \frac{C_\tau + B_\tau}{M} \le \frac{C_\tau + 6P_\tau M}{M} \le 6\left(\frac{C_\tau}{M} + P_\tau\right) \le 6\overline{X}'_\tau$.

Assume now that $P_\tau < \frac{1}{2}$. Then $\mathbf{E}[B_\tau] \le M/2$. Again, using Chernoff bound, $\mathbf{Pr}[B_\tau \ge 3M] \le 2^{-3M}$. But if $B_\tau \le 3M$, then $\overline{Y}_\tau = \frac{C_\tau + B_\tau}{M} \le \frac{C_\tau}{M} + 3 \le 8\overline{X}'_\tau$, since $\overline{X}'_\tau \ge \max\left\{\frac{1}{2}, \frac{C_\tau}{M}\right\}$.

In either case, $\overline{Y}_\tau \le 8\overline{X}'_\tau$ with probability at least $1 - 2^{-3M}$. Using the union bound over all $\tau \in I$ and the fact that $M \ge 100 \log N$ while $|I| \le N^2$ gives the desired result. □

The final solution is obtained by scaling the values $y(v)$ up by the factor of 8.

D Integrality Gap for Directed Layered Graphs

We sketch the lower bound of $\Omega(\log n)$ on the integrality gap of the LP for RMFC on directed layered graphs due to Khanna and Olver [12]. Let d be a parameter, and $\lambda = \lceil e^d \rceil$. The construction has $\lambda + 1$ layers $L_0, \ldots, L_\lambda$ where $L_0 = \{s\}$, and L_τ contains τd vertices for all $\tau \ge 1$. The terminal set is $T = L_\lambda$. For each $0 \le \tau \le \lambda - 1$, there is an edge (u, v) for every $u \in L_\tau, v \in L_{\tau+1}$.

Consider the following fractional solution: for each $v \in L_\tau$, we set $x(v) = 1/(\tau d)$. It is easy to see that this is a feasible fractional solution: for each terminal $t \in T$, for each path $P \in \mathcal{P}_t$, we have $\sum_{u \in P} x(u) = \frac{1}{d} \sum_{\tau \le \lambda} \frac{1}{\tau} \ge \frac{1}{d} \ln \lambda \ge 1$. The total LP-weight of each layer is exactly 1, so the cost of the solution is 1. It is easy to see that the cost of the optimal integral solution for this instance is d. Since the construction size $n = d(\sum_{\tau \le \lambda} \tau) = O(d\lambda^2) = O(de^{2d})$, the integrality gap is $d = \Omega(\log n)$.