Firefighting on Trees: $(1 - 1/e)$–Approximation, Fixed Parameter Tractability and a Subexponential Algorithm*

Leizhen Cai[1], Elad Verbin[2], and Lin Yang[1]

[1] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong SAR, China
{lcai,lyang}@cse.cuhk.edu.hk
[2] The Institute For Theoretical Computer Science, Tsinghua University, Beijing, China
elad.verbin@gmail.com

Abstract. The firefighter problem is defined as follows. Initially, a fire breaks out at a vertex r of a graph G. In each subsequent time unit, a firefighter chooses a vertex not yet on fire and protects it, and the fire spreads to all unprotected neighbors of the vertices on fire. The objective is to choose a sequence of vertices for the firefighter to protect so as to save the maximum number of vertices. The firefighter problem can be used to model the spread of fire, diseases, computer viruses and suchlike in a macro-control level.

In this paper, we study algorithmic aspects of the firefighter problem on trees, which is NP-hard even for trees of maximum degree 3. We present a $(1-1/e)$-approximation algorithm based on LP relaxation and randomized rounding, and give several FPT algorithms using a random separation technique of Cai, Chan and Chan. Furthermore, we obtain an $2^{O(\sqrt{n}\log n)}$-time subexponential algorithm.

1 Introduction

The Firefighter problem is a one-person's game on a graph G defined as follows. At time $t = 0$, a fire breaks out at a vertex r of G. For each time step $t \geq 1$, a firefighter protects one vertex not yet on fire (the vertex remains protected thereafter), and then the fire spreads from burning vertices (i.e., vertices on fire) to all unprotected neighbors of these vertices. The process ends when the fire can no longer spread, and then all vertices that are not burning are considered *saved*. The objective is to choose a sequence of vertices for the firefighter to protect so as to save the maximum number of vertices in the graph. The Firefighter problem was introduced by Hartnell [Har95] in 1995 and can be used to model the spread of fire, diseases, computer viruses and suchlike in a macro-control level.

The Firefighter problem is NP-hard even for trees of maximum degree 3 as shown by Finbow et al. [FKMR07]. On the other hand, Hartnell and Li [HL00]

* Partially supported by Earmarked Research Grant 410206 of the Research Grants Council of Hong Kong SAR, China.

S.-H. Hong, H. Nagamochi, and T. Fukunaga (Eds.): ISAAC 2008, LNCS 5369, pp. 258–269, 2008.

have proved that a simple greedy method for trees is a 0.5-approximation algorithm, and MacGillivray and Wang [MW03] have solved the problem in polynomial time for some special trees. Various aspects of the problem have been considered by Develin and Hartke [DH07], Fogarty [Fog03], Wang and Moeller [WM02], and Cai and Wang [CW07], among others. We refer the reader to a recent survey of Finbow and MacGillivray [FM07] for more information on the Firefighter problem.

In this paper, we study algorithmic aspects of the Firefighter problem on trees. Our main results are:

1. A $(1 - 1/e)$-approximation algorithm (Section 3) based on a LP-relaxation and randomized rounding. We also prove that $1 - 1/e$ is the best approximation factor one can get using any LP-respecting rounding technique with the same LP (Section 3.1).
2. Several FPT algorithms and polynomial-size kernels (Section 4 and also a summary in Table 1) when we use several different choices for the parameter k: the number of saved vertices, the number of saved leaves, and the number of protected vertices. Our FPT algorithms are based on the random separation method of Cai, Chan and Chan [CCC06].
3. A subexponential algorithm (Section 5) that runs in time $2^{O(\sqrt{n}\log n)}$. We note that an $2^{O(n^{0.33})}$-time algorithm would falsify a conjecture that there is no subexponential algorithm for SAT (see discussions in Section 5).

Table 1. Summary of FPT algorithms. The "randomized complexity" column indicates expected running time of algorithms with one-sided error, and the "deterministic complexity" column gives the worst-case running time of deterministic algorithms.

problems	randomized	deterministic	kernel size
Saving k Vertices	$O(4^k + n)$	$O(n) + 2^{O(k)}$	$O(k^2)$
Saving All But k Vertices	$O(4^k n)$	$2^{O(k)} n \log n$	open
Saving k Leaves	$O(n) + 2^{O(k)}$	$O(n) + 2^{O(k)}$	$O(k^2)$
Saving All But k Leaves	none unless $NP \subseteq RP$	NP-complete for $k = 0$	no kernel
Maximum k-Vertex Protection	$k^{O(k)} n$	$k^{O(k)} n \log n$	open

2 Definitions and Notation

We first define some terms. Let T be a rooted tree with root r which is the origin of the fire. A vertex is *protected* once it is protected by the firefighter, and *saved* if it is not burnt at the end of the game. A *strategy* for the Firefighter problem is a sequence $v_1, v_2, \ldots, v_t$ of protected vertices of T such that vertex $v_i, 1 \leq i \leq t$, is protected at time i and the fire can no longer spread to unprotected vertices at time t.

The following is the decision version of the problem we consider in the paper.

Firefighter on Trees

Instance A rooted tree T with root r and a positive integer k.

Question Is there a strategy for the firefighter to save at least k vertices when a fire breaks out at the root r?

Without ambiguity, we abbreviate "the Firefighter problem on trees" as "the Firefighter problem" in the rest of the paper.

We denote the subtree of T rooted at vertex v by $T(v)$, and assign the number of vertices in $T(v)$ as the weight w_v of v. For a strategy S, the *value* of S, denoted by $\|S\|$, equals the number of vertices saved by S.

Denote by L_i the set of vertices of depth i (L_0 contains only the root), and we refer to each L_i as a *level* of T. Let d_v denote the depth of v, and h denote the height of T, i.e., the depth of the deepest leaf. We write $u \preceq v$ if u is an ancestor of v.

An instance of a *parameterized problem* consists of a pair (I, k) with I being the input and k the parameter, and a parameterized problem is *fixed-parameter tractable* (FPT in short) if it admits an *FPT algorithm*, i.e., an algorithm that runs in $f(k)\,|I|^{O(1)}$ time for some computable function f independent of the input size $|I|$. A *kernelization* for a parameterized problem is a polynomial-time reduction that maps an instance (I, k) onto (I', k') such that (1) $|I'| \leq g(k)$ for some computable function g, (2) $k' \leq k$, and (3) (I, k) is a "Yes"-instance iff (I', k') is a "Yes"-instance. The pair (I', k') is called a *kernel* of (I, k). The kernel is *polynomial-size* if $g(k) = k^{O(1)}$. The existence of a kernel implies the existence of an FPT. The existence of an FPT implies the existence of a kernel, but not necessarily of a polynomial-size kernel.

3 A $(1 - 1/e)$–Approximation Algorithm

In this section we present a $(1-1/e)$-approximation algorithm for the Firefighter problem on trees, which improves the 1/2-approximation of Hartnell and Li [HL00] (note $(1 - 1/e) \approx 0.6321$). Our algorithm, proposed by B. Alspach (see [FM07]), uses randomized rounding of an LP relaxation of a 0-1 integer program formulated by MacGillivray and Wang [MW03]. It is asked in [FM07] to investigate the performance of this algorithm, and in this section we determine the approximation ratio of the algorithm.

It is easy to see that an optimal strategy for a tree protects a vertex at level i at time i and has no need to protect descendants of a protected vertex. This observation translates into the following 0-1 integer program of MacGillivray and Wang [MW03] for a tree $T = (V, E)$, where for vertex v, x_v is a boolean decision variable such that $x_v = 1$ iff v is protected, and w_v is the number of descendants of v.

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{v \in V} w_v x_v & \\
\text{subject to} & x_r = 0 & \\
& \displaystyle\sum_{v \in L_i} x_v \leq 1 & \text{for every level } L_i \text{ with } i \geq 1 \\
& \displaystyle\sum_{v \preceq u} x_v \leq 1 & \text{for every leaf } u \text{ of } T \\
& x_v \in \{0, 1\} & \text{for every vertex } v \text{ of } T
\end{array}
\tag{1}
$$

By relaxing the constraint $x_v \in \{0,1\}$ in the above integer program to $0 \leq x_v \leq 1$, we get a linear program, whose optimal solution will be denoted by OPT_{LP}. The optimal solution to the LP can be interpreted as an optimal "fractional" firefighting strategy[1].

Alspach's rounding method uses the fact that in OPT_{LP}, the values of x_v in each level of the tree sum up to at most 1, and thus they can be treated as a probability distribution. The rounding scheme is to pick the vertex to protect at each level according to this distribution. It might be the case that the fractional values in a level sum up to less than 1. In this case, with the remaining probability we choose to protect no vertex at the level (this makes no difference in the analysis). Also, it might be the case that the rounding procedure chooses to protect both a vertex v and its ancestor u. In this case, we choose to protect u rather than v, and do not protect any vertex in v's level. We call this situation an *annihilation*.

We note that the loss in rounding stems exactly from annihilations. If annihilations never occur, the expected value of the rounded strategy is at least $\|OPT_{LP}\|$ and thus the approximation ratio would be 1. On the other hand, if annihilations occur, consider a vertex v which is fully saved by the fractional strategy and consider the path of length d_v from the root r to v. In the worst case, the fractional strategy assigns a $1/d_v$-fraction of a firefighter to each vertex in this path. In this case, the probability that v is saved by the rounded strategy is equal to

$$1 - (1 - 1/d_v)^{d_v} \geq 1 - 1/e \ .$$

To turn this intuition into a full analysis, we just need to show that the above case is indeed the worst case, and that a similar behavior occurs when v is not fully saved by the fractional strategy. We do this in the following lemma.

Lemma 1. *Given any fractional strategy S_F, let S_I be the integer strategy produced by applying the randomized rounding method to S_F. Then,*

$$\mathbf{E}\left[\|S_I\|\right] \geq \left(1 - \frac{1}{e}\right) \cdot \|S_F\| \ .$$

Proof. Denote the value of the fractional strategy at v by $\tilde{x}_v$ and the value of the rounded strategy at v by x_v. Thus, x_v is an indicator random variable for whether v is protected by S_I. Similarly, for each v, define $\tilde{y}_v = \sum_{u \preceq v} \tilde{x}_v$ to be the fraction of v that is saved by S_F, and $y_v = \sum_{u \preceq v} x_v$ to indicate whether v is saved by S_I.

[1] A fractional firefighting strategy is a placement of fractional firefighters on the vertices so that the sum of firefighter fractions assigned to each level is at most 1, and the sum of firefighter fractions assigned to each root-to-leaf path is at most 1. For example, if a vertex v is protected by half of a firefighter, all its descendants are half-saved. If, furthermore, another vertex $u \succ v$ is protected by 0.3 of a firefighter, then all descendants of u are 0.8-saved.

Fix a vertex v, and denote by $r = v_0, v_1, v_2, \ldots, v_k = v$ the path from the root to v. By the definition of the rounding procedure, we see that

$$\mathbf{Pr}\left[y_v = 1\right] = 1 - \prod_{i=1}^{k}(1 - \tilde{x}_{v_i}).$$

We have the following bound

$$\begin{aligned}\mathbf{Pr}\left[y_v = 1\right] &= 1 - \prod_{i=1}^{k}(1 - \tilde{x}_{v_i}) \geq 1 - \left(\frac{\sum_{i=1}^{k}(1 - \tilde{x}_{v_i})}{k}\right)^k = 1 - \left(\frac{k - \sum_{i=1}^{k}\tilde{x}_{v_i}}{k}\right)^k \\ &= 1 - \left(\frac{k - \tilde{y}_v}{k}\right)^k = 1 - \left(1 - \frac{\tilde{y}_v}{k}\right)^k \geq 1 - e^{-\tilde{y}_v} \geq \left(1 - \frac{1}{e}\right)\tilde{y}_v,\end{aligned}$$

where the first inequality follows from the inequality of the means, and the second and third inequalities follow from standard analysis, using the fact that $0 \leq \tilde{y}_v \leq 1$. Note that the sum of all $\tilde{y}_v$ is just the value of S_F. Therefore,

$$\mathbf{E}\left[\|S_I\|\right] \;=\; \sum_{v \in V}\mathbf{Pr}\left[y_v = 1\right] \;\geq\; \sum_{v \in V}\left(1 - \frac{1}{e}\right)\tilde{y}_v \;=\; \left(1 - \frac{1}{e}\right) \cdot \|S_F\|,$$

where the first equality follows from linearity of expectation. This finishes the proof of the lemma. □

The above lemma implies that the expected approximation ratio of our algorithm is $(1 - 1/e)$. We can easily derandomize our algorithm by using the method of conditional expectations [AS92], which will be discussed in the full paper.

Theorem 1. *There is a deterministic polynomial-time* $(1 - 1/e)$*-approximation algorithm for the firefighter problem on trees.*

3.1 LP-Respecting Rounding Does Not Achieve Approximation Better Than $1 - 1/e$

We note that the best known integrality gap for MacGillivray and Wang's LP, proved by Hartke [Har06], is $\frac{16}{17}$. Thus, it is tempting to believe that the rounding method might be improvable. However, we can show that no rounding technique from a relatively rich class of rounding techniques gives an approximation ratio better than $1 - 1/e$. This means that one would have to try something very different than standard rounding methods.

A common feature of many rounding techniques in the literature is that they are *LP-respecting*. A rounding technique for the firefighter problem is called LP-respecting if it only chooses to protect vertices v with $\tilde{x}_v > 0$, and never protects any vertex v with $\tilde{x}_v = 0$. (Recall that $\tilde{x}_v$ is the value of the optimal LP solution on vertex v.) The following theorem states that *any* LP-respecting rounding technique, when used together with [MW03]'s LP, does not achieve an approximation ratio better than $1 - 1/e$. Consequently, any rounding technique that aims at getting better than $(1 - 1/e)$–approximation would have to be *LP-disrespecting*.

Theorem 2. *For any $\epsilon > 0$, there exists a tree T, and an optimal fractional solution $\tilde{S}$ for the LP on T, such that any integral strategy $S \subseteq \{v : \tilde{x}_v > 0\}$ can save no more than $(1 - 1/e + \epsilon) \cdot \|OPT_{IP}(T)\|$ vertices.*

The proof of this theorem is somewhat technical, and we defer it to the full version of the paper.

4 FPT Algorithms

In this section, we consider FPT algorithms and polynomial-size kernels for three parameterized versions of the Firefighter problem.

1. Saving k Vertices: The parameter k is the number of saved vertices, and we ask if there is a strategy saving at least k vertices.
2. Saving k Leaves: The parameter k is the number of saved leaves, and we wish to determine if the firefighter can save at least k leaves of the tree.
3. Maximum k-Vertex Protection: The parameter k is the number of protected vertices and we wish to find a strategy for the firefighter to protect k vertices to maximize the total number of saved vertices.

The results in this section are summarized in Table 1.

The main tool we use is the *random separation* method of Cai, Chan, and Chan [CCC06]. This technique produces randomized algorithms, which can be derandomized by using *universal sets* (see [NSS95]). A set of binary vectors of length n is (n, t)-*universal* if for every subset of size t of the indices, all 2^t configurations appear in the set. Naor et al. [NSS95] give a construction of a (n, t)-universal set of cardinality $2^t t^{O(\log t)} \log n$ in time $2^t t^{O(\log t)} n \log n$.

4.1 Saving k Vertices

First we use random separation to give an $2^{O(k)}n$-time algorithm for Saving k Vertices. Then we construct a kernel of size $O(k^2)$ for the problem. Finally we use the random separation method to solve the parametric dual problem Saving All But k Vertices in time $2^{O(k)}n$.

We start with our FPT algorithm for Saving k Vertices. Call a strategy S *satisfying* if it saves at least k vertices. The goal is then to find a satisfying strategy if one exists. First observe that if the root r has a child v with weight $w_v \geq k$ then we can protect v to solve the problem. Therefore we can assume from now on that the weight of every vertex in $V - r$ is at most $k - 1$. In particular, this means that the height of T is at most k.

The algorithm first colors each vertex of T randomly and independently by either green or red with equal probability. We call a coloring of T *good* if T has a satisfying strategy S_0 such that all vertices in S_0 are green, and all descendants of vertices in S_0 are red.

Given a good coloring of T, we can find a satisfying strategy S to T as follows:

Step 1. Find the set V_g of green vertices whose descendants are all red.
Step 2. For each level L_i ($i \geq 1$), choose from $V_g \cap L_i$ a vertex of maximum weight, and put it in S. (If $V_g \cap L_i = \emptyset$ then do nothing).

It's not hard to see that S is a satisfying strategy, as for each vertex v of S_0, v is in $V_g \cap L_{d_v}$ and can be placed in S, thus $\|S\| \geq \|S_0\|$. Therefore S is indeed a satisfying strategy, and we can find it in $O(n)$ time, given a good coloring.

However, the probability of obtaining a good coloring depends on the sum of the number of vertices in S_0 and the number of the descendants of S_0, which might be as large as $\Theta(k^2)$. We can reduce this sum to $2k$ by using the following simple existence lemma:

Lemma 2. *Suppose that every non-trivial subtree of T is of size less than k, and that there is a satisfying strategy. Then there is a satisfying strategy S_0 that saves at most $2k$ vertices.*

Proof. Let S_1 be some satisfying strategy. If S_1 saves at most $2k$ vertices, then we are done. Otherwise, we construct S_0 as follows. Since $w_v < k$ for each vertex v of S_1, we can add vertices from S_1 to S_0 in turn until S_0 saves at least k vertices. This S_0 saves at most $2k$ vertices. □

Thus, if we choose S_0 not as an arbitrary satisfying strategy, but as the satisfying strategy guaranteed in Lemma 2, then the probability that a randomly-chosen coloring is good is at least $1/2^{2k}$. By choosing 4^k colorings and running steps 1 and 2 for each of the colorings, we succeed in finding a satisfying strategy, if one exists, with at least constant probability.

To derandomize the algorithm, we can use a $(n, 2k)$-universal set, and use the vectors of the set as the colorings. If a satisfying strategy exists, then at least one of the colorings will be good. Therefore we have a deterministic FPT algorithm that runs in time $4^k k^{O(\log k)} n \log n$.

We now present an $O(k^2)$-size kernel for Saving k Vertices on trees. First, note that if r has a child v with $w_v \geq k$, we can protect v to save at least k vertices. In such a case, the problem is solvable in time $O(n)$, and we can use a trivial kernel for it. From now on we assume that the weight of each child of r is at most $k-1$ and in particular the height h of T is at most $k-1$.

The idea behind the kernel is to ignore all but a limited number of vertices in each level of the tree. Its construction is as follows:

Step 1. If level 1 has at least k vertices then we put the k largest-weight vertices of level 1 into K_1 else we put all vertices of level 1 into K_1.
Step 2. For $i := 2$ to h, if level i has at least $2k - i$ vertices then we put the $2k - i$ largest-weight vertices of level i into K_i else we put all vertices of level i into K_i.
Step 3. The kernel is $K = \bigcup_{i=1}^{h} K_i$.

Note that the above construction of K can be performed in time $O(n)$. Also note that K has at most $k + \sum_{i=2}^{h}(2k - i) \leq 3k^2/2$ vertices. We prove that K is a kernel in the following lemma:

Lemma 3. *If T has a satisfying strategy S, then there exists a satisfying strategy $S' \subseteq K$.*

Proof. Let $S = \{v_1, v_2, \ldots, v_t\}$ be a satisfying strategy, where v_i is in level i. Define $I(S)$ to be the largest i such that $v_i \notin K_i$ (define $I(S) = \infty$ if $S \subseteq K$). We say that S is *minimal* iff the removal of any vertex from S would make S non-satisfying. We let S be a satisfying strategy that maximizes $I(S)$. Furthermore, among all of those we choose S to be a strategy which is minimal. Let $i = I(S)$. If $i = \infty$, we are finished. We assume $i < \infty$, and eventually reach a contradiction.

We will now show how to replace vertex v_i of S by a vertex in K_i to get another satisfying strategy. If $i = 1$ then $|K_1| = k$ as $v_1 \notin K_1$. Note that at most $k-1$ vertices in K_1 are ancestors of vertices in S as $|S| \leq k$. Therefore K_1 has a vertex v that is not an ancestor of any vertex of S. Since $w_v \geq w_{v_1}$, we can remove v_1 from S and insert v, to get a satisfying strategy S' of T with $i(S') > i(S)$, contradicting the choice of S. Otherwise $i > 1$. Since $v_i \notin K_i$, we have $|K_i| = 2k-i$. Note that at most $k-i$ vertices in K_i are ancestors of vertices in S as $|S| \leq k$. Furthermore, by the minimality of S, at most $k-1$ vertices in K_i are descendants of vertices in S (otherwise S does not need vertex v_i). Therefore K_i has at least one vertex v that is neither an ancestor nor a descendent of any vertex in S. By the definition of K_i, $w_v \geq w_{v_i}$ and we can replace v_i in S by v to get a satisfying strategy S' of T with $i(S') > i(S)$, contradicting the choice of S. □

The $O(k^2)$ kernel can be easily combined with the FPT algorithm to establish the following result.

Theorem 3. *Saving k Vertices can be solved in $O(n) + 4^k k^{O(\log k)}$ time.*

For the parametric dual Saving All But k Vertices of Saving k Vertices, we can also use random separation to obtain an FPT algorithm that runs in $2^{O(k)} n \log n$ time. The main difference is that we use a random coloring to "guess" the burnt part instead of the saved part for Saving k Vertices. The details will be given in the full paper.

4.2 Saving k Leaves and Protecting k Vertices

In this section, we consider FPT algorithms for Saving k Leaves and Maximum k-Vertex Protection. The former uses the number of saved leaves as the parameter k, and the latter tries to save the maximum number of vertices by protecting k vertices.

We start with Saving k Leaves, which deals with the situation that leaves are much more valuable than internal vertices. Due to space limit, we will only sketch the main ideas of our FPT algorithm and leave the details to the full paper. Note that the parametric dual Saving All But k Leaves of Saving k Leaves is NP-complete even for $k = 0$, which was shown by Finbow et. al. [FKMR07]. Thus Saving All But k Leaves has no FPT algorithm unless $P \neq NP$.

To solve Saving k Leaves, it is possible to use the algorithm in the latter part of this section for Maximum k-Vertex Protection, which takes $k^{O(k)} n$ time. Here we

describe an algorithm that takes time $2^{O(k)}poly(n)$. To get such an algorithm, it is tempting to try the same approach we used for the Saving k Vertices problem, but such an approach does not work: for Saving k Vertices we used the fact that if there is a strategy that saves at least k vertices, then there is a strategy whose subtrees are of total size $O(k)$. This does not hold for Saving k Leaves.

The main difficulty comes from the *snakes* in the tree. A *snake* is a path $(v_1, v_2, \ldots, v_\ell)$ with $\ell \geq 2$, such that for each $i = 1, \ldots, \ell - 1$, v_i has only a single child, v_{i+1}. (In other words, a snake is an induced path). If snakes do not exist, then a random separation algorithm similar to the one we gave for Saving k Vertices solves Saving k Leaves in time $2^{O(k)}n$. Clearly, the difficulty lies in dealing with snakes. To this end, we use a random separation approach together with an algorithm for finding a maximum matching, which is used to decide which vertex to protect inside each snake.

The first step of the algorithm is to contract each snake, and to get a snake-free tree T'. We then apply the random separation method on T' to find a satisfying (saving at least k leaves) strategy S'. Then we somehow transform S' into a satisfying strategy S of T. Note that every vertex v' in T' corresponds to either a snake or a single vertex in the original tree T, and to save the leaves that v' saves, we can protect any vertex in the snake that corresponds to v'. With this observation we can understand the transformation from S' to S as a scheduling problem, with the vertices in S' being the set of tasks, their corresponding snakes indicating the starting time and the deadline of the tasks, and each level in T being a free slot in the processing queue. We formulate this problem into a bipartite graph and find the satisfying strategy using the maximal matching algorithm for bipartite graphs. We will discuss details in the full paper.

We can use an approach similar to the one used for Saving k Vertices to obtain an $O(k^2)$-size kernel of Saving k Leaves, which can be combined with the above FPT algorithm to obtain the following result:

Theorem 4. *Saving k Leaves can be solved in $O(n) + 2^{O(k)}$ time.*

We now turn to Maximum k-Vertex Protection, the problem of protecting k vertices to save the maximum number of vertices. Note that for any tree of height k, an optimal strategy needs only protect at most k vertices. Therefore Maximum k-Vertex Protection can be also regarded as a parameterized version of the firefighter problem on trees when we take the height of a tree as the parameter k.

Theorem 5. *Maximum k-Vertex Protection can be solved in $k^{O(k)}n$ time by a randomized algorithm.*

Proof. We color the vertices randomly and independently, with probability $\frac{1}{k}$ to be green and probability $1 - \frac{1}{k}$ to be red. Let S_0 be an assumed optimal strategy. We call a coloring c *good* if all vertices in S_0 are green and all of their ancestors are red. If the coloring is good, then we can find an optimal strategy by the same procedure that we used in the algorithm for Saving k Vertices, in time $O(n)$. Now, since there are k vertices in S_0, and each of them has at most k ancestors, the probability for the coloring to be good is at least

$$\geq \left(\frac{1}{k}\right)^k \cdot \left(\frac{k-1}{k}\right)^{k^2} \geq \left(\frac{1}{k}\right)^k \left(\frac{1}{4}\right)^k = k^{-O(k)}$$

Thus, picking $k^{O(k)}$ colorings allows us to find the optimal strategy with probability at least a constant. □

This algorithm cannot be derandomized by using normal universal sets while maintaining the running time, because we color each vertex green or red with unequal probabilities. We can use equal probability for green and red and then use a (n, k^2)-universal set to derandomize the algorithm, but the resulting algorithm runs in time of $2^{O(k^2)} n \log n$. To derandomize the above algorithm efficiently, Verbin [Ver] has recently introduced *asymmetric universal sets* which can be used to yield a deterministic algorithm that runs in time $k^{O(k)} n \log n$.

Question 1. *Is there an algorithm for* Maximum k-Vertex Protection *that runs in time* $2^{o(k \log k)} poly(n)$*?*

5 A Subexponential Algorithm

In this section we present an algorithm for exactly solving the firefighter problem on trees. The algorithm takes time $n^{O(\sqrt{n})} = 2^{O(\sqrt{n} \log n)}$ on any tree with n vertices.

The main idea is to use pruning, coupled with a careful analysis of the size of the space of feasible solutions. We note that the size of the space of feasible solutions can be $2^{\Omega(n)}$, and an exhaustive approach is clearly not sufficient. On the other hand, an exhaustive approach is good enough when the tree is somewhat balanced, and in particular if the tree is of height $O(\sqrt{n})$. Our algorithm deals with non-balanced trees by detecting parts of the tree that are small and "costly to save", and pruning such parts. By "costly to save", we mean that there are vertices on the same levels such that if we protect them, we will save many more vertices.

We now present the algorithm FF-SUBEXP, which operates recursively, and solves the firefighter problem in time $n^{O(\sqrt{n})}$. Recall that r denotes the root of T, w_v denotes the number of vertices in the subtree rooted at v, d_v denotes the depth of v, and *level* i refers to the set of all vertices of depth i. We set the parameter $k_0 = \sqrt{n}$, which is fixed throughout the execution of the algorithm, even when we call the algorithm recursively.

FF-SUBEXP works as follows. Its input is T, a tree with n vertices. Its output is an optimal firefighter strategy for T.

1. If $n \leq k_0$, run a brute-force search (taking time $O(n^{k_0})$) and return the result it gives. Otherwise, continue to step 2.
2. If r has some child v with $w_v \leq k_0$, then:
   (a) Construct a tree T' which is identical to T except that the subtree rooted at v is completely deleted. Run FF-SUBEXP recursively on T' to get an optimal strategy S' for T'.

(b) Calculate the best strategy S'' out of the strategies that protect one vertex in each of the levels 1 through w_v, and no vertices below level w_v. Do this using the naive brute-force algorithm that takes time $O(n^{w_v+1})$.
(c) Pick the strategy among S' and S'' which, when run on T, gives the best result. Return it. /* we will prove that this strategy is optimal in the full paper */

3. else: /* all children of the root are subtrees of size at least k_0. Here we'll do a sort of brute-force search */
   (a) Go over all children $v_1, \ldots, v_\ell$ of r. For each v_i do:
      – Find the best strategy, S_i, among all strategies that protect v_i. Do this by recursively running FF-SUBEXP on a tree with $n - w_{v_i} - \ell$ vertices, produced by deleting v_i's subtree, deleting all level-1 nodes, and making all level-2 nodes into direct descendants of r.
   (b) Pick the strategy among $S_1, \ldots, S_\ell$ that gives the best result when applied to T, and return it.

Due to the space constraint, we will defer the correctness proof and complexity analysis of the algorithm to the full paper.

It is interesting to note that the NP-hardness reduction of Finbow, King, MacGillivray and Rizzi in [FKMR07] in fact implies that an $2^{O(n^{0.33})}$-time algorithm for the firefighter problem on trees would imply an $2^{o(n)}$-time algorithm for solving 3-SAT on instances with n variables and $O(n)$ clauses. This would falsify a conjecture of Impagliazzo et al. [IP01, IPZ01]. Furthermore, the reduction of [FKMR07] also implies that an $n^{O(k^{0.99})}$-time algorithm for the firefighter problem on trees of height k would falsify the same conjecture of Impagliazzo et al. . Recall that the trivial implementation of step 2b in the algorithm takes time $O(n^k)$, which means that the implementation of that step, although naive, is likely to be close to optimal.

Question 2. *Is there an exact $2^{n^{1/3}polylog(n)}$-time algorithm for the firefighter problem on trees? Alternatively, would an $2^{n^{1/2}/polylog(n)}$-time algorithm for the problem imply an $2^{o(n)}$-time algorithm for 3-SAT with n variables and $O(n)$ clauses?*

References

[AS92] Alon, N., Spencer, J.H.: The Probabilistic Method. Wiley, Chichester (1992)

[CCC06] Cai, L., Chan, S.M., Chan, S.O.: Random separation: A new method for solving fixed-cardinality optimization problems. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 239–250. Springer, Heidelberg (2006)

[CW07] Cai, L., Wang, W.: The surviving rate of a graph (manuscript, 2007)

[DH07] Develin, M., Hartke, S.G.: Fire containment in grids of dimension three and higher. Discrete Appl. Math. 155(17), 2257–2268 (2007)

[FKMR07] Finbow, S., King, A., MacGillivray, G., Rizzi, R.: The firefighter problem for graphs of maximum degree three. Discrete Mathematics 307(16), 2094–2105 (2007)

[FM07] Finbow, S., MacGillivray, G.: The firefighter problem: a survey (manuscript, 2007)

[Fog03] Fogarty, P.: Catching the Fire on Grids, M.Sc. Thesis, Department of Mathematics, University of Vermont (2003)

[Har06] Hartke, S.G.: Attempting to narrow the integrality gap for the firefighter problem on trees. In: Discrete Methods in Epidemiology. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 70, pp. 179–185 (2006)

[Har95] Hartnell, B.: Firefighter! An application of domination. In: 24th Manitoba Conference on Combinatorial Mathematics and Computing, University of Minitoba, Winnipeg, Cadada (1995)

[HL00] Hartnell, B., Li, Q.: Firefighting on trees: how bad is the greedy algorithm? Congr. Numer. 145, 187–192 (2000)

[IP01] Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. 62(2), 367–375 (2001)

[IPZ01] Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. 63(4), 512–530 (2001)

[MW03] MacGillivray, G., Wang, P.: On the firefighter problem. J. Combin. Math. Combin. Comput. 47, 83–96 (2003)

[NSS95] Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: IEEE Symposium on Foundations of Computer Science, pp. 182–191 (1995)

[Ver] Verbin, E.: Asymmetric universal sets (in preparation)

[WM02] Wang, P., Moeller, S.: Fire control on graphs. J. Combin. Math. Combin. Comput. 41, 19–34 (2002)