Journal of Computer and System Sciences 80 (2014) 1285–1297

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Parameterized complexity of firefighting ☆

Cristina Bazgan [a,b], Morgan Chopin [a], Marek Cygan [c,*], Michael R. Fellows [d], Fedor V. Fomin [e], Erik Jan van Leeuwen [e]

[a] Université Paris-Dauphine, LAMSADE, France
[b] Institut Universitaire de France, France
[c] Institute of Informatics, University of Warsaw, Poland
[d] Charles Darwin University, Australia
[e] Department of Informatics, University of Bergen, Norway

ARTICLE INFO

Article history:
Received 24 November 2011
Received in revised form 31 January 2014
Accepted 3 March 2014
Available online 15 March 2014

Keywords:
Firefighter problem
Fixed parameter tractability
Kernelization

ABSTRACT

The Firefighter problem is to place firefighters on the vertices of a graph to prevent a fire with known starting point from lighting up the entire graph. In each time step, a firefighter may be placed on an unburned vertex, permanently protecting it, and the fire spreads to all neighboring unprotected vertices of burning vertices. The goal is to let as few vertices burn as possible. In this paper, we consider a generalization of this problem, where at each time step $b \geqslant 1$ firefighters can be deployed. Our results answer several open questions raised by Cai et al. [8]. We show that this problem is W[1]-hard when parameterized by the number of saved vertices, protected vertices, and burned vertices. We also investigate several combined parameterizations for which the problem is fixed-parameter tractable. Some of our algorithms improve on previously known algorithms. We also establish lower bounds to polynomial kernelization.

1. Introduction

The Firefighter problem concerns a deterministic model of fire spreading through a graph via its edges. The problem has recently received considerable attention [13,18]. In the model, we are given a graph G with a vertex $s \in V(G)$. At time $t = 0$, the fire breaks out at s and vertex s starts burning. At each step $t \geqslant 1$, first the firefighter protects one vertex not yet on fire — this vertex remains permanently protected — and the fire then spreads from burning vertices to all unprotected neighbors of these vertices. The process stops when the fire cannot spread anymore. The goal is to find a strategy for the firefighter that minimizes the amount of burned vertices, or, equivalently, maximizes the number of saved, i.e. not burned, vertices.

It is known that the Firefighter problem is NP-hard, even when restricted to bipartite graphs [18] or trees of maximum degree three [14]. However, it is polynomial-time solvable on such trees if the root — the initially burning vertex — has degree two [18]. We refer to the survey [15] for an overview of further combinatorial results on the problem.

In this paper, we consider the parameterized complexity of the Firefighter problem on general graphs and trees where, at each time step, b firefighters can be deployed. Denote by b-Firefighter the generalization of the Firefighter problem

☆ The preliminary results of this publication were presented at IPEC 2011 [10] and ISAAC 2011 [2].

* Corresponding author.
E-mail addresses: bazgan@lamsade.dauphine.fr (C. Bazgan), chopin@lamsade.dauphine.fr (M. Chopin), cygan@mimuw.edu.pl (M. Cygan), michael.fellows@cdu.edu.au (M.R. Fellows), fedor.fomin@ii.uib.no (F.V. Fomin), erikjan@mpi-inf.mpg.de (E.J. van Leeuwen).

http://dx.doi.org/10.1016/j.jcss.2014.03.001
0022-0000/

with b firefighters. For any fixed budget $b \geqslant 2$, b-Firefighter was proved NP-hard for trees of maximum degree $b+3$, and polynomial-time solvable on trees of maximum degree $b+2$ provided that the fire breaks out at a vertex of degree at most $b+1$ [3].

The study of the problem from the perspective of parameterized complexity was initiated by Cai, Verbin, and Yang [8] for the case $b=1$. They considered the following three parameterized versions of the problem and obtained a number of parameterized algorithms on trees.

The first parameterization considered by Cai et al. in [8] is by the number of saved vertices.

Saving k-Vertices **Parameter:** k
Input: An undirected graph G, a vertex s, and two integers k and b.
Question: Is there a strategy with respect to the budget b to save at least k vertices when a fire breaks out at s?

Cai et al. proved that Saving k-Vertices on trees has a polynomial kernel. They also gave a randomized algorithm solving Saving k-Vertices on trees in time $O(4^k+n)$, which can be derandomized to an $O(n+2^{O(k)})$-time algorithm.

The second parameterization considered by Cai et al. is by the number of burned vertices.

Saving All But k-Vertices **Parameter:** k
Input: An undirected graph G on n vertices, a vertex s, and two integers k and b.
Question: Is there a strategy with respect to the budget b to save at least $n-k$ vertices when a fire breaks out at s?

For Saving All But k-Vertices on trees, Cai et al. gave a randomized algorithm of running time $O(4^k n)$, which can be derandomized to an $O(2^{O(k)} n \log n)$-time algorithm. They left as an open problem whether Saving All But k-Vertices has a polynomial kernel on trees.

The third parameterization investigated by Cai et al. is by the number of protected vertices, that is, the total number of vertices occupied by firefighters.

Maximum k-Vertex Protection **Parameter:** k
Input: An undirected graph G, a vertex s, and two integers k and b.
Question: A strategy with respect to the budget b that saves the maximum number of vertices by protecting a total of at most k vertices when a fire breaks out at s?

For Maximum k-Vertex Protection on trees, Cai et al. gave a randomized algorithm of running time $O(k^{O(k)} n)$, which can be derandomized to an $O(k^{O(k)} n \log n)$-time algorithm. They left open whether the problem has a polynomial kernel on trees, and asked whether there is an algorithm solving the problem on trees in time $2^{o(k \log k)} n^{O(1)}$.

We will sometimes consider the decision variant of Maximum k-Vertex Protection.

k-Vertex Protection **Parameter:** k
Input: An undirected graph G, a vertex s, and three integers k, b and a.
Question: Is there a strategy with respect to the budget b that saves at least a vertices by protecting a total of at most k vertices when a fire breaks out at s?

The unparameterized version of this problem is obviously NP-hard on trees of maximum degree three from the hardness of the Firefighter problem with $b=1$, and NP-hard on trees of maximum degree $b+3$ for any fixed $b \geqslant 2$ from the hardness of the b-Firefighter problem.

Our results. We resolve several open questions of Cai, Verbin, and Yang [8]. We also refine and extend some of the results of [8]. Fig. 1 summarizes our results.

	Saving k-Vertices	k-Vertex Protection	Saving All But k-Vertices
k	• *W[1]-hard* • **XP** • **FPT for planar graphs**	• *W[1]-hard* • **XP**	• **W[1]-hard** • **XP**
Poly Kernel?	*no*	*no*	*no*
k + b	• **W[1]-hard** • *XP*	• **W[1]-hard** • *XP*	• **FPT**
Poly Kernel?	*no*	*no*	**no**
k + tw	• **FPT**	• **FPT**	• open
Poly Kernel?	open	**no**	**no**
k + τ	• **FPT**	• *FPT*	• **FPT**
Poly Kernel?	open	open	open

Fig. 1. Summary of our results for general graphs. Results in bold font are proved in this paper, and directly imply the results in italics. The vertex cover number is denoted by τ, and the treewidth by tw. Recall that $\tau \geqslant tw$.

- In Section 2, we first observe that on general graphs the problem is W[1]-hard for any budget $b \geqslant 1$, which was independently observed by Cai (private communication) for $b = 1$. The problem is shown to be in FPT when parameterized by k and the treewidth of a graph. We also give a deterministic algorithm solving Saving k-Vertices on trees in time $O((b+1)^{k/b+3}kn)$, improving the running time $O(4^k + n)$ of the randomized algorithm from [8] for $b = 1$. We further derive that Saving k-Vertices is in FPT on graphs of bounded local treewidth, including planar graphs, graphs of bounded genus, apex-minor-free graphs, and graphs of bounded maximum vertex degree.
- In Section 3, we first establish that on general graphs the problem is W[1]-hard. We provide deterministic algorithms solving Saving All But k-Vertices in time $O((b+1)^k n)$ on trees, and in time $O((2^{b+1}-1)^{k+b-1}n)$ on general graphs. Consequently the problem becomes FPT when parameterized by $k + b$. The algorithm on trees improves the $O(4^k n)$ running time of the randomized algorithm from [8] for $b = 1$. We also answer the open question of Cai et al. by showing that Saving All But k-Vertices has no polynomial kernel on trees of maximum vertex degree four for $b = 1$, and no polynomial kernel on trees of maximum vertex degree $b + 4$ for any $b \geqslant 2$.
- For Maximum k-Vertex Protection, we answer two open questions of Cai et al.: We give a deterministic algorithm solving Maximum k-Vertex Protection on trees in time $O((b+1)^{k/b+1}kn)$ in Section 2, and show that the problem has no polynomial kernel on trees in Section 3. The no-poly-kernel result was independently obtained by Yang [20]. Based on the parameterized algorithm, we also give an exact subexponential-time algorithm, solving the b-Firefighter problem on an n-vertex tree in time $O((b+1)^{\sqrt{2n/b}}n^{3/2})$, thus improving on the $2^{O(\sqrt{n}\log n)}$ running time from [8,16] for $b = 1$. On general graphs, we show that the Maximum k-Vertex Protection problem is W[1]-hard, but is in FPT when parameterized by k and the treewidth of a graph.

Graph terminology. All graphs in this paper are undirected, connected, finite, and simple. Let $G = (V, E)$ be a graph. The *open* (resp. *closed*) *neighborhood* of a vertex $v \in V$ is the set $N(v) = \{u \in V : (u, v) \in E\}$ (resp. $N[v] = N(v) \cup \{v\}$). Given a subset $S \subseteq V$, the open (resp. closed) neighborhood of S is the set $N(S) = \bigcup_{u \in S} N(u) \setminus S$ (resp. $N[S] = N(S) \cup S$). We denote by $d_G(u, v)$ the minimum length of a path in G with endpoints $u, v \in V$. Finally, let $G \setminus X$ be the graph induced by $V \setminus X$ where $X \subseteq V$. Throughout this paper, the vertex cover number of G will be denoted by $\tau(G)$ or τ and the treewidth of G by $tw(G)$ or tw.

Parameterized complexity. Here we only give the basic notions on parameterized complexity; for more background the reader is referred to [11]. Parameterized complexity is a framework which provides a new way to express the computational complexity of problems. A problem parameterized by k is called *fixed-parameter tractable* (fpt) if there exists an algorithm, called an fpt-algorithm, that solves it in time $f(k)n^{O(1)}$ (fpt-time). The function f is typically super-polynomial, and depends only on k. In other words, the combinatorial explosion is confined into f. A parameterized problem P with parameter k will be denoted by (P, k). The class XP is the set of parameterized problems (P, k) that can be solved in time $n^{g(k)}$ for a given computable function g.

One of the main tools to design fpt-algorithms is *kernelization*. A kernelization algorithm transforms in polynomial time an instance I of a given problem parameterized by k into an equivalent instance I' of the same problem parameterized by $k' \leqslant k$ such that $|I'| \leqslant g(k)$ for some computable function g. The instance I' is called a *kernel* of size $g(k)$ — if g is a polynomial then I' is a *polynomial kernel*. By applying any, even exponential, algorithm to a kernel of a given problem, we can derive an fpt algorithm for that problem.

We can also give evidence that an fpt-algorithm does not exist for a certain problem, i.e., indicating the parameterized intractability of a problem. To this end, we need to introduce the notion of a *parameterized reduction*. An fpt-reduction is an algorithm that reduces any instance I of a problem with parameter k to an equivalent instance I' with parameter $k' = g(k)$ in fpt-time for some function g. The basic class of parameterized intractability is W[1] and there is a good reason

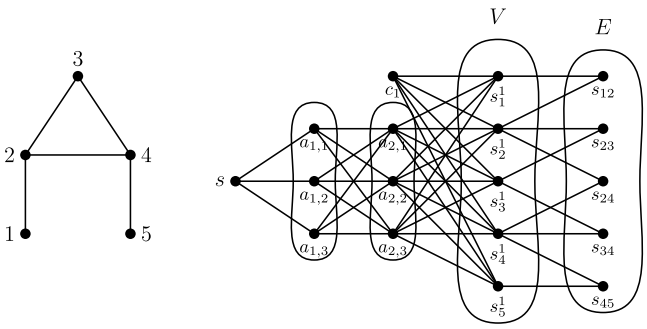

Fig. 2. An instance of k-Clique and the corresponding graph G' constructed in the proof of Theorem 2.1 for $k = 3$ and $b = 1$.

to believe that W[1]-hard problems — hard according to the fpt-reduction — are unlikely to be in FPT. We have the following inclusions: FPT $\subseteq$ W[1] $\subseteq$ XP.

2. Saving and protecting vertices

In this section, we consider the complexity of Saving k-Vertices and Maximum k-Vertex Protection. These problems are known to be fixed-parameter tractable on trees, but their complexity on general graphs was hitherto unknown. We resolve this open problem by giving a W[1]-hardness result for both problems. We also improve the algorithms known to exist for trees. At the other end of the spectrum, we extend the boundary where Saving k-Vertices and Maximum k-Vertex Protection remain fixed-parameter tractable by giving parameterized algorithms on graphs of bounded treewidth.

2.1. W[1]-hardness on bipartite graphs

We show that Saving k-Vertices and the decision variant of Maximum k-Vertex Protection are W[1]-hard, even on bipartite graphs. We reduce from the k-Clique problem, which is well known to be W[1]-hard [11].

Theorem 2.1. Saving k-Vertices *is W[1]-hard even on bipartite graphs for any budget* $b \geqslant 1$.

Proof. Let (G, k) be an instance of k-Clique. We construct the following bipartite graph G' (see Fig. 2). For each edge $(u, v) \in E(G)$, we add a vertex s_{uv}; this set of vertices is denoted by E. Add b copies $V^1, \ldots, V^b$ of $V(G)$, i.e., for each vertex $v \in V(G)$, we add vertices $s_v^1 \in V^1, \ldots, s_v^b \in V^b$. Now add an edge from s_{uv} to both s_u^h and s_v^h for each $(u, v) \in E(G)$ and each $h = 1, \ldots, b$. Add a root vertex s, and add vertices $a_{i,j}$ for all $1 \leqslant i \leqslant k-1$ and $1 \leqslant j \leqslant (k-1)b+1$. Connect $a_{i,j}$ to $a_{i',j'}$ $(i' = i+1)$ for all i, j, j', connect $a_{1,j}$ to s for all j, and connect $a_{k-1,j}$ to each vertex of $V' = \bigcup_{1 \leqslant h \leqslant b} V^h$ for all j. Finally, add b vertices $c_1, \ldots, c_b$ adjacent to all vertices of V'. Now set $k' = kb + \binom{k}{2} + b$.

We claim that Saving k-Vertices on (G', s, k', b) is a Yes-instance if and only if k-Clique on (G, k) is a Yes-instance. Suppose that G has a k-clique K. Then the strategy that protects the vertices $s_v^1, \ldots, s_v^b$ for all $v \in K$ saves the vertices s_{uv} for all $u, v \in K$. Since K is a clique, these vertices s_{uv} are indeed present in G'. Additionally, we can protect (and thus save) vertices $c_1, \ldots, c_b$. It follows that this strategy saves at least k' vertices.

Suppose that $S = \{p_1^1, \ldots, p_1^b, \ldots, p_\ell^1, \ldots, p_\ell^b\}$ is a strategy for (G', s, k', b) that chooses vertices $p_t^1, \ldots, p_t^b$ at time t and saves at least k' vertices. First observe that if $p_t^h = a_{i,j}$ for some i, j, and h, then this vertex is not helpful, as there is always a vertex $a_{i,j'}$ that will be burned at time t and has the same neighborhood as $a_{i,j}$. Hence we can assume that no vertex $a_{i,j}$ is protected by the strategy. This implies that all vertices of V' will be burned, except those that are protected by the strategy. But then protecting vertices of E does not save any further vertices. Since the fire will reach V' in k time steps, and thus E in $k+1$ time steps, the vertices in $S \cap V'$ are responsible for saving $\binom{k}{2}$ vertices, which is only possible if the vertices of $S \cap V'$ induce a k-clique in G. □

Observe that essentially the same construction works for the decision variant of Maximum k-Vertex Protection.

Theorem 2.2. k-Vertex Protection *is W[1]-hard even on bipartite graphs for any budget* $b \geqslant 1$.

Proof. We again reduce from k-Clique and construct the same bipartite graph as in the proof of Theorem 2.1. We set $k' = kb + b$, $a' = kb + \binom{k}{2} + b$. Correctness now follows straightforwardly from the arguments in the proof of Theorem 2.1. □

The above reduction is also an NP-hardness reduction, and simpler than the original reduction for the Firefighter problem on bipartite graphs [18].

2.2. Algorithms on general graphs

We show that Saving k-Vertices and the decision variant of Maximum k-Vertex Protection are both in XP.

Before we present the algorithms, we need to consider a different version of the b-Firefighter problem, where in each round an arbitrary number of vertices may be protected under the following restrictions:

- each protected vertex must have a neighbor which is on fire,
- after i rounds of the process at most ib vertices are protected.

By Saving k-Vertices II we denote the Saving k-Vertices problem where vertices are protected subject to the above rules. The problem k-Vertex Protection II is defined equivalently.

Lemma 2.3. *An instance (G, s, k, b) of the* Saving k-Vertices *problem is a* Yes*-instance if and only if it is a* Yes*-instance of the* Saving k-Vertices II *problem. This equivalence also holds for* k-Vertex Protection *and* k-Vertex Protection II.

Proof. We prove the result for Saving k-Vertices and Saving k-Vertices II. Assume that (G, s, k, b) is a Yes-instance of the Saving k-Vertices problem. Let P be the set of protected vertices of an optimum strategy S. We construct a strategy S', which in the i-th round of Saving k-Vertices II protects exactly those vertices of P which have a neighbor which is on fire. Clearly after i rounds at most ib vertices will be protected, since each vertex of P is protected by the strategy S' not earlier than by the strategy S.

In the other direction, assume that (G, s, k, b) is a Yes-instance of the Saving k-Vertices II problem and P is the set of protected vertices of an optimum strategy S'. We construct a strategy S as follows. Let $(v_1, \ldots, v_{|P|})$ be a sequence of vertices of P sorted by the round in which a vertex is protected by S' (breaking ties arbitrarily). In the i-th round of strategy S we protect the vertices $v_{(i-1)b+1}, \ldots, v_{ib}$. The vertex v_j, $j \in \{(i-1)b+1, \ldots, ib\}$, is not on fire in the i-th round, because in the strategy S' it is protected not earlier than in the i-th round.

The proof for showing the equivalence between k-Vertex Protection and k-Vertex Protection II is similar. □

Lemma 2.4. *Let $G = (V, E)$ be a graph with an initially burned vertex $s \in V$. Verifying if a subset $S \subseteq (V \setminus \{s\})$ is a valid strategy with respect to the budget b for* k-Vertex Protection II *can be done in linear time.*

Proof. Let $L_i = \{v \in V\colon d_{G \setminus \bigcup_{0 \leqslant j \leqslant i-1} L_j \cap S}(s, v) = i\}$ for any $i > 0$ and $L_0 = \{s\}$. The graph $G \setminus \bigcup_{0 \leqslant j \leqslant i-1} L_j \cap S$ is obtained from G by removing protected vertices from time step 1 through $i-1$. Let $r_i = ib - |\bigcup_{0 \leqslant j < i} L_j \cap S|$ be the number of available firefighters in round i. If there exists $i \in \{1, \ldots, k\}$ such that $|S \cap L_i| > r_i$ or $r_i < 0$, then this strategy is not valid. □

Lemma 2.5. Maximum k-Vertex Protection II *is solvable in time $n^{O(k)}$.*

Proof. Let $(G = (V, E), s, k, b)$ be an instance of Maximum k-Vertex Protection II. Among all strategies $S \subseteq V$ such that S is a valid strategy and $|S| \leqslant k$, the algorithm simply chooses the strategy that saves the largest number of vertices. From Lemma 2.4, the running time is $n^{O(k)}$. □

Lemma 2.6. Saving k-Vertices II *is solvable in time $n^{O(k)}$.*

Proof. Let (G, s, k, b) be an instance of Saving k-Vertices II. We run the above algorithm for Maximum k-Vertex Protection II for all $k' = 1, \ldots, k$. Observe that it is possible to save k vertices of the graph if and only if the algorithm saves at least k vertices for some value of k'. This implies a running time of $n^{O(k)}$. □

Using Lemmas 2.3, 2.5 and 2.6 we get the following two theorems.

Theorem 2.7. Saving k-Vertices *is solvable in time $n^{O(k)}$.*

Theorem 2.8. Maximum k-Vertex Protection *is solvable in time $n^{O(k)}$.*

Notice that the parameters in the reductions used in Theorem 2.1 and Theorem 2.2 are quadratically related. Since k-Clique cannot be solved in time $n^{o(k)}$ unless FPT = M[1] [9], we obtain the following lower bounds.

Corollary 2.9. Saving k-Vertices *cannot be solved in time $n^{o(\sqrt{k})}$, unless* FPT = M[1].

Corollary 2.10. Maximum k-Vertex Protection *cannot be solved in time $n^{o(\sqrt{k})}$, unless* FPT = M[1].

2.3. Improved algorithm on trees

We show that Saving k-Vertices and Maximum k-Vertex Protection have a deterministic $O((b+1)^{k/b+3}kn)$ and $O((b+1)^{k/b+1}kn)$ algorithm, respectively, on trees. This resolves an open question of Cai et al. [8] for $b=1$. As a consequence, we also obtain a refined subexponential algorithm for the Firefighter problem on trees, running in time $O(2^{\sqrt{2n}}n^{3/2})$.

The following observation is a straightforward adaptation of the one by MacGillivray and Wang [18, Sect. 4.1].

Lemma 2.11. *For any optimum strategy for an instance of the b-Firefighter problem on trees, there is an integer ℓ such that all protected vertices have depth at most ℓ, exactly b vertices $p_i^1,\dots,p_i^b$ at each depth $1\leqslant i\leqslant \ell-1$ are protected, and $b'\leqslant b$ vertices $p_\ell^1,\dots,p_\ell^{b'}$ at depth ℓ are protected. Moreover, all ancestors of each p_i^h are burned.*

We need the following notation. Let T be any rooted tree. Use a pre-order traversal of T to number the vertices of T from 1 to n. We say that $u\in V(T)$ is *to the left* of $v\in V(T)$ if the number assigned to u is not greater than the number of v in the order. It is then easy to define what the *leftmost* or *rightmost* vertex is.

Theorem 2.12. Maximum k-Vertex Protection *on trees is solvable in time $O((b+1)^{(k/b)+1}kn)$.*

Proof. Let (T,s,k,b) be an instance of Maximum k-Vertex Protection on a tree T. Assume that T is rooted at s and let $h=\lceil k/b\rceil$. By Lemma 2.11, we can define a characteristic vector χ_v of length h for each vertex v of the tree, which has value $p_i\in\{0,\dots,b\}$ at position i if and only if the optimal strategy protects p_i vertices at depth i in the part of the tree to the left of v. We use these vectors as the basis for a dynamic programming procedure. However, the vector cannot ensure that no ancestors of a protected vertex will be protected. To ensure this, we add another dimension to our dynamic programming procedure. The pre-order numbering ensures that no descendant is protected.

The dynamic programming algorithm is then as follows. Let L be the set of vertices in T that are at depth at most h. For each $v\in L$, let P_v denote the path in T between v and s. For each vector $\chi\in\{0,\dots,b\}^h$ and each integer $0\leqslant i\leqslant h$, we compute $A_v(\chi,i)$, the maximum number of vertices one can save when protecting at most $\chi(j)$ vertices at depth j, where protected vertices must lie to the left of v but at depth greater than i when lying on P_v, and no protected vertex is an ancestor of another. Observe that s is the leftmost vertex of L. Now set $A_s(\chi,i)=0$ for any χ and i. Then

$$A_v(\chi,i)=\max\Big\{A_{l(v)}\big(\chi,\min\{\operatorname{depth}(v)-1,i\}\big),\\ \big[\chi\big(\operatorname{depth}(v)\big)\geqslant 1\wedge\operatorname{depth}(v)>i\big]\big(r(v)+A_{l(v)}\big(\chi^v,\operatorname{depth}(v)-1\big)\big)\Big\}$$

Here $\operatorname{depth}(v)$ is the depth of a vertex v, $l(v)$ is the rightmost vertex in L which has strictly smaller value in the pre-order than v, and $r(x)$ is the number of vertices saved when protecting only x. Moreover, χ^v is the vector obtained from χ by reducing the number $p_{\operatorname{depth}(v)}$ by 1. In the formula we use Iverson's bracket notation, where $[\phi]$ is equal to one if ϕ is true and zero otherwise.

To see that the above formula is correct, observe that we can either protect the considered vertex v or not. If we do not protect v, then we must ensure that the value for the second dimension of our dynamic programming procedure does not exceed the length of P_v, yet still captures the same forbidden part of P_v. Correctness then follows from the fact that the parent of v is always on $P_{l(v)}$. If we do protect v, we can protect v only if we are allowed to do so, i.e. if $\chi(\operatorname{depth}(v))\geqslant 1$ and $\operatorname{depth}(v)>i$. Furthermore, we need to ensure that no ancestor of v is protected later. Therefore, we set the value for the second dimension of our dynamic programming procedure to $\operatorname{depth}(v)-1$.

To get the solution for the whole tree T, return $A_{v^*}(\chi^*,0)$, where v^* is the rightmost vertex of L and χ^* is a vector of length h with the h-th entry set to $k-(h-1)b$ and the other entries set to b. To obtain the claimed running time, first find L, and then $l(v)$ for each vertex $v\in L$. This can be done in linear time by a depth-first search. We can also compute $r(x)$ for each $x\in V(T)$ in linear time, as $r(x)$ equals one plus the number of descendants of x. By traversing the vertices of L from left to right, the total running time is $O((b+1)^h kn)=O((b+1)^{(k/b)+1}kn)$. □

Corollary 2.13. Saving k-Vertices *on trees is solvable in $O((b+1)^{(k/b)+3}kn)$ time.*

Proof. Let (T,s,k,b) be an instance of Saving k-Vertices on a tree T. Using the same argument from Lemma 2.6, we run the above algorithm for Maximum k-Vertex Protection for all $k'=1,\dots,k$. Furthermore, we note that

$$\sum_{i=1}^{k}\big((b+1)^{(i/b)+1}in\big)\leqslant kn(b+1)\sum_{i=1}^{k}(b+1)^{i/b}\leqslant(b+1)^{k/b+3}kn,$$

implying that the worst-case running time is $O((b+1)^{k/b+3}kn)$. □

To obtain a good subexponential algorithm for the b-Firefighter problem, we use the following lemma. A similar idea for $b = 1$ appeared independently in [16].

Lemma 2.14. *If a vertex at depth d burns in an optimum strategy for an instance of the b-Firefighter problem on trees, then at least $\frac{b}{2}(d^2 + d)$ vertices are saved.*

Proof. Let (T, s, b) be an instance of the b-Firefighter problem on trees, and let v be a vertex of depth d that burns in an optimum strategy. Then the strategy protects b vertices at depth d, and by Lemma 2.11 it thus protects b vertices $p_i^1, \ldots, p_i^b$ at each depth i for $1 \leqslant i \leqslant d$. For any i, each of the subtrees rooted at $p_i^1, \ldots, p_i^b$ should contain at least $d - i + 1$ vertices, or it would have been better to protect the vertex at depth i that is on the path from v to s. But then the strategy saves at least $b \sum_{i=1}^{d} (d - i + 1) = \frac{b}{2}(d^2 + d)$ vertices. □

Theorem 2.15. *b-Firefighter on trees is solvable in $O((b+1)^{\sqrt{2n/b}} n^{3/2})$ time.*

Proof. Let (T, s, b) be an instance of the b-Firefighter problem on trees. Suppose that a vertex v at depth $\sqrt{2n/b}$ burns in an optimum strategy. Then, by Lemma 2.14, the strategy saves at least $n + \sqrt{bn/2} > n$ vertices, which is not possible. It follows that all vertices at depth $\sqrt{2n/b}$ are saved in any optimum strategy. Since in any optimum strategy every protected vertex has a burned ancestor by Lemma 2.11, all protected vertices are at depth at most $\sqrt{2n/b}$. Hence there is an optimum strategy that protects at most $b\sqrt{2n/b} = \sqrt{2bn}$ vertices, and we can find the optimum strategy by running the algorithm of Theorem 2.12 with $k = \sqrt{2bn}$. □

This implies an $O(2^{\sqrt{2n}} n^{3/2})$-time algorithm for Firefighter on trees.

2.4. Tractability on graphs of bounded treewidth

We generalize the above results by showing that Maximum k-Vertex Protection and Saving k-Vertices remain fixed-parameter tractable when parameterized by k and the treewidth of the underlying graph. To this end, we use Monadic Second Order Logic (MSOL). The syntax of MSOL of graphs includes the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, variables for vertices, edges, sets of vertices, and sets of edges, the quantifiers $\forall$, $\exists$ that can be applied to these variables, and the following four binary relations:

1. $u \in U$, where u is a vertex variable and U is a vertex set variable.
2. $d \in D$, where d is an edge variable and D is an edge set variable.
3. $\mathrm{adj}(u, v)$, where u, v are vertex variables, and the interpretation is that u and v are adjacent.
4. Equality, $=$, of variables representing vertices, edges, sets of vertices, and sets of edges.

For Maximum k-Vertex Protection, we actually need Linear Extended MSOL [1], which allows the maximization over a linear combination of the size of unbound set variables in the MSOL formula. (The definition of LEMSOL in [1] is slightly more general, but this suffices for our purposes.)

Theorem 2.16. *Maximum k-Vertex Protection is solvable in $f(t,k)n^{O(1)}$ time, where t is the treewidth of the given graph.*

Proof. Let (G, s, k, b) be an instance of Maximum k-Vertex Protection such that the treewidth of G is t. We may assume that $b \leqslant k$. Use Bodlaender's Algorithm [4] to find a tree decomposition of G of width at most t. Consider the following MSOL formulae.

$$\begin{aligned}&\mathrm{NextBurn}\big(B_{i-1}, B_i, p_1^1, \ldots, p_1^{b_1}, \ldots, p_i^1, \ldots, p_i^{b_i}\big)\\ &:= \forall v \left(\left(v \in B_{i-1} \vee \exists u \left(u \in B_{i-1} \wedge \mathrm{adj}(u, v) \wedge \left(\bigwedge_{\substack{1 \leqslant j \leqslant i \\ 1 \leqslant h \leqslant b_j}} v \neq p_j^h \right) \right) \right) \Leftrightarrow v \in B_i \right)\end{aligned}$$

This expresses is that if the vertices of B_{i-1} are burning by time step $i-1$, then the vertices of B_i burn by time step i, assuming that vertices $p_1^1, \ldots, p_1^{b_1}, \ldots, p_i^1, \ldots, p_i^{b_i}$ have been protected so far, with $b_j \leqslant b$ for $j = 1, \ldots, i$.

$$\begin{aligned}&\mathrm{Saved}\big(S, B, p_1^1, \ldots, p_1^{b_1}, \ldots, p_m^1, \ldots, p_m^{b_m}\big)\\ &:= \forall u \left(u \in S \Rightarrow \left(u \notin B \wedge \forall v \left(\mathrm{adj}(u, v) \Rightarrow v \in S \vee \bigvee_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant h \leqslant b_i}} p_i^h = u \right) \right) \right)\end{aligned}$$

This expresses that S is a set of saved vertices when B is a set of burned vertices and vertices $p_1^1, \ldots, p_1^{b_1}, \ldots, p_m^1, \ldots, p_m^{b_m}$ are protected, with $b_j \leqslant b$ for $j = 1, \ldots, m$.

$$\begin{aligned}
\text{Protect}(S, b_1, \ldots, b_\ell) := {} & \exists p_1^1, \ldots, p_1^{b_1}, \ldots, p_\ell^1, \ldots, p_\ell^{b_\ell}\ \exists B,\ B_0, \ldots, B_{\ell-1} \quad \forall u\ (u \in B_0 \Leftrightarrow u = s) && (1)\\
& \wedge \bigwedge_{1 \leqslant i \leqslant \ell-1} \text{NextBurn}\big(B_{i-1}, B_i, p_1^1, \ldots, p_1^{b_1}, \ldots, p_i^1, \ldots, p_i^{b_i}\big) && (2)\\
& \wedge \bigwedge_{\substack{1 \leqslant i \leqslant \ell \\ 1 \leqslant h \leqslant b_i}} p_i^h \notin B_{i-1} && (3)\\
& \wedge \forall u \left(\left(\bigvee_{0 \leqslant i \leqslant \ell-1} u \in B_i \right) \Rightarrow u \in B \right) && (4)\\
& \wedge \text{Saved}\big(S, B, p_1^1, \ldots, p_1^{b_1}, \ldots, p_\ell^1, \ldots, p_\ell^{b_\ell}\big) && (5)
\end{aligned}$$

This expresses that S can be saved by protecting b_i vertices in round i. The sets B_i contain all vertices that are burned by time step i, which is ensured by the formulas in lines (1) and (2). The set B contains vertices that are not saved (line (5)) and all vertices of the sets B_i (line (4)). The vertices $p_1^1, \ldots, p_1^{b_1}, \ldots, p_\ell^1, \ldots, p_\ell^{b_\ell}$ are the vertices that are protected. Line (3) ensures that the vertices we want to protect are not burned by the time we pick them. Then we want to find the largest set S such that

$$\text{Protect}_k(S) := \bigvee_{1 \leqslant \ell \leqslant \lceil k/b \rceil} \ \bigvee_{\substack{1 \leqslant d \leqslant b \\ b(\ell-1)+d \leqslant k}} \text{Protect}\big(S, \overbrace{b, \ldots, b}^{\ell-1}, d\big)$$

is true. Following a result of Arnborg, Lagergren, and Seese [1], this can be done in $f(k,t) \cdot n^{O(1)}$ time using the above formula. □

In the same way as Corollary 2.13, we then obtain the following.

Corollary 2.17. Saving k-Vertices *is in FPT when parameterized by k and the treewidth of the graph.*

Observe that this algorithm also works on graphs of bounded local treewidth, because if the graph has a vertex at distance more than k from the root, then any strategy that protects a vertex at distance i from the root in time step i will save at least k vertices, and we can answer Yes immediately.

Corollary 2.18. Saving k-Vertices *is in FPT on graphs of bounded local treewidth.*

The class of graphs having bounded local treewidth coincides with the class of apex-minor-free graphs [12], which includes the class of planar graphs.

Corollary 2.19. Saving k-Vertices *is in FPT on planar graphs.*

2.5. Kernelization feasability

In this section, we provide a kernelization for Saving k-Vertices when parameterized by τ and k.

Theorem 2.20. Saving k-Vertices *admits a kernel of size at most $O(2^\tau k)$.*

Proof. Let (G, s, k, b) be an instance of Saving k-Vertices. A set $T \subseteq V$ is called a *twins set* if for every $v, u \in T$, $v \neq u$, we have $N(u) = N(v)$ and $(u, v) \notin E$. Consider the following reduction rule.

Rule: If there exists a twins set T such that $|T| \geqslant k+1$, then delete $|T| - k$ vertices of T.

Let $G' = (V', E')$ be the graph obtained by iteratively applying the above rule to every twins set in G. Notice that the procedure runs in polynomial time. Let $C \subseteq V'$ be a minimum vertex cover and let $D = V' \setminus C$ be an independent set. The number of distinct twins sets in D is at most 2^τ (one for each subset of C). Moreover, each twins set in G' has at most k vertices. Therefore, the size of the reduced instance is at most $O(2^\tau k)$.

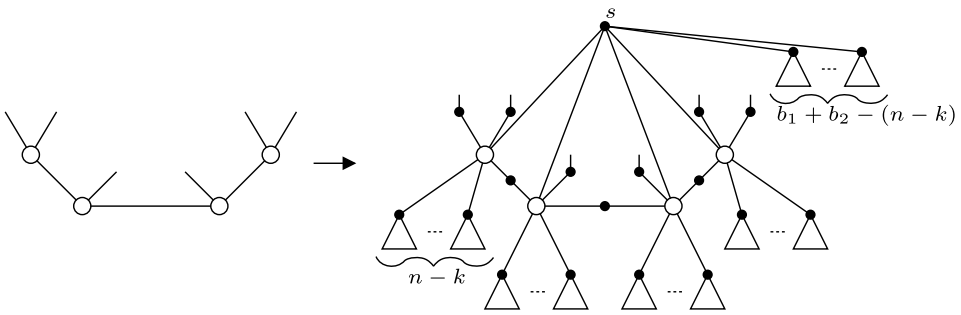

Fig. 3. The construction of G' from G in the proof of Theorem 3.1. Added vertices are black and a triangle represents a busy gadget.

Correctness of the rule. We have the following observation.

Observation: *Let $G = (V, E)$ be a graph, $T \subseteq V$ be a twins set, and S be a strategy that saves at least k vertices. If S protects a subset $T_1 \subseteq T$ then protecting any subset $T_2 \subseteq T$ instead of T_1 such that $|T_2| = |T_1|$ leads to another strategy S' that saves exactly the same number of vertices.*

Suppose that there exists a strategy S that saves at least k vertices in G. It follows from the above observation that if S protects a vertex in a twins set that has been deleted by the reduction rule then we can protect any other non-deleted vertex in the same twins set instead, without changing the solution value. Correctness follows from the fact that it is always possible to apply the previous observation to obtain a strategy S' that saves at least k vertices in G'. Conversely, if a strategy saves at least k vertices in G', then this strategy clearly saves at least k vertices in G. □

3. Burning vertices

In this section, we consider the b-Firefighter problem when parameterized by the number of burned vertices, which we call the Saving All But k-Vertices problem. We first establish that this problem is W[1]-hard even on bipartite graphs. We improve on results of Cai et al. [8] by showing an $O((b+1)^k n)$-time deterministic algorithm for trees and an $O((2^{b+1}-1)^k n)$-time deterministic algorithm for general graphs. Furthermore, we prove that the Saving All But k-Vertices problem has no polynomial kernel for trees, resolving an open problem from [8].

3.1. W[1]-hardness on bipartite graphs

We show that Saving All But k-Vertices is W[1]-hard, even on bipartite graphs. We reduce from the following W[1]-hard problem [7].

Regular k-Clique **Parameter:** k
Input: A d-regular graph $G = (V, E)$ and an integer k.
Question: Is there a k-clique in G?

Theorem 3.1. Saving All But k-Vertices *is W[1]-hard even on bipartite graphs.*

Proof. In this reduction, a *busy* gadget denotes a $(b+k)$-star with center c (i.e., a tree with one internal vertex c and $b+k$ leaves). Attaching a busy gadget to a vertex v means to create a copy of a $(b+k)$-star and to make c adjacent to v. Thus, if v is burning at a given time step, then c has to be protected; otherwise more than k vertices would burn.

Let (G, k) be an instance of Regular k-Clique. We construct the following bipartite graph G' from the d-regular graph G. Add a new vertex s adjacent to all vertices of G. Attach $b_1 + b_2 - (n-k)$ busy gadgets to s where $b_1 = k(n-k)$ and $b_2 = kd - \binom{k}{2}$. Attach $n-k$ busy gadgets to every vertex in V. Remove every edge $(u, v) \in E$ and add an edge-vertex x_{uv} adjacent to u and v (see Fig. 3). Now set $k' = k+1$ and $b = b_1 + b_2$. Notice that, at time step 1, there are only $n-k$ firefighters that can be placed freely because of the busy gadgets.

We claim that Saving All But k-Vertices on (G', s, k', b) is a Yes-instance if and only if Regular k-Clique on (G, k) is a Yes-instance. Suppose that we have a k-clique K and consider the following strategy. At time step one, the strategy uses the $n-k$ remaining firefighters to protect all the original vertices V in G' except those in K. At time step two, all the k vertices of K are burned. Since there are $n-k$ busy gadgets attached to each vertex in K, we need to protect $b_1 = k(n-k)$

vertices. Moreover, there are $kd - \binom{k}{2}$ edge-vertices adjacent to the vertices in the k-clique. Since there remain $b_2 = b - b_1$ firefighters, we can protect all of them. Hence, no more than k vertices are burned at the end of the process.

Conversely, suppose that there is no k-clique in G. At time step 1, any valid strategy has to place the $n - k$ remaining firefighters on vertices that are not edge-vertices; otherwise at least $k' + 1$ vertices will burn. At time step two, since there is no k-clique, there will be at least $kd - \binom{k}{2} + 1$ edge-vertices adjacent to the k burned vertices. For the same reason as before, there remains $b_2 = b - b_1$ firefighters which is not enough to protect these edge-vertices. Therefore, given any valid strategy there will be at least k' burned vertices. □

3.2. Algorithms on general graphs

In this subsection, we first show that Saving All But k-Vertices can be solved in $n^{O(k)}$. We also show that this algorithm is essentially optimal, unless FPT = M[1]. Next, we show an $O((2^{b+1} - 1)^k n)$-time algorithm on general graphs for any $b \geqslant 1$.

We first define, as in Section 2.2, the Saving All But k-Vertices II problem that can be proved equivalent to Saving All But k-Vertices as in Lemma 2.3. Next, we introduce the notion of *valid burning set*. Given an instance $(G = (V, E), s, k, b)$ of Saving All But k-Vertices II, a valid burning set is a subset $B \subseteq V$ with $s \in B$ such that there exists a strategy for which, at the end of the process, the burned vertices are exactly those in B.

Lemma 3.2. *Let $G = (V, E)$ be a graph with an initially burned vertex $s \in V$. Verifying if a subset $B \subseteq V$ is a valid burning set can be done in linear time.*

Proof. Observe that the set of protected vertices must be exactly $N(B)$. For any $v \in N(B)$, let $d(v)$ be the length of a shortest s-v path in G whose internal vertices are all in B. Then v has to be protected before or at time step $d(v)$. It follows that B is a valid burning set if and only if the number of vertices v for which $d(v) \leqslant t$ is at most bt for every $t = 1, \ldots, k$. Finally, we note that $d(v)$ can be easily computed using a breadth-first search. Hence we can determine whether B is a valid burning set in linear time. □

Theorem 3.3. Saving All But k-Vertices *is solvable in time $n^{O(k)}$.*

Proof. Exhaustively consider all subsets $B \subseteq V$ with $|B| \leqslant k$. If B is a valid burning set, then the answer is Yes. If no valid burning set is found, then the answer is No. It follows from Lemma 3.2 that the running time is $n^{O(k)}$. □

Notice that the parameters in the reduction used in Theorem 3.1 are linearly related. Since Regular k-Clique cannot be solved in time $n^{o(k)}$ unless FPT = M[1] [19], we obtain the following lower bound that shows that the algorithm given in Theorem 3.3 is optimal.

Corollary 3.4. Saving All But k-Vertices *cannot be solved in time $n^{o(k)}$, unless* FPT = M[1].

Theorem 3.5. Saving All But k-Vertices II *on general graphs is solvable in $O((2^{b+1} - 1)^{k+b-1} n)$ time and polynomial space.*

Proof. We present a simple branching algorithm. Assume that we are in the i-th time step and let B be the set of vertices which are currently on fire. Moreover, let P be the set of already protected vertices (in the first round we have $B = \{s\}$ and $P = \emptyset$). Let $a = ib - |P|$ and $r = |N(B) \setminus P|$. The algorithm does the following:

1. If $|B| > k$, then we immediately answer No.
2. Observe that in the i-th round we are allowed to protect at most $\min(a, r)$ vertices. If $a \geqslant r$, then we can greedily protect the whole set $N(B) \setminus P$. Hence in this case we answer Yes.
3. In the last case, when $a < r$, we branch on all subsets of $N(B) \setminus P$ of size at most a. Observe that the number of branches is equal to $\sum_{j=0}^{a} \binom{r}{j} \leqslant 2^r - 1$, since we have $a < r$.

The running time of the algorithm is as follows. We introduce a measure $\alpha = (k - |B|) + (ib - |P|)$ which we use in our time bound. At the beginning of the first round of the burning process, we have $\alpha = (k - 1) + (b - 0) = k + b - 1$. By $T(\alpha)$ we denote the upper bound on the number of steps that our algorithm requires for a graph with measure value α. Observe that for $\alpha \leqslant 0$, we have $T(\alpha) = O(n)$. Let us assume that the algorithm did not stop in step 1 nor 2, and it branches into at most $2^r - 1$ choices of protected vertices. Observe that no matter how many vertices the algorithm decides to protect, the value of the measure decreases by exactly $r - b$. Consequently, we have the inequality $T(\alpha) \leqslant (2^r - 1)T(\alpha - r + b) + O(n)$. Since the algorithm did not stop in steps 1 or 2, we infer that $r \geqslant b + 1$. The time bound follows from the fact that the worst case for the inequality occurs when $r = b + 1$. □

Using Lemmas 2.3 and 3.5 we deduce the following result.

Corollary 3.6. Saving All But k-Vertices *on general graphs is solvable in* $O((2^{b+1}-1)^{k+b-1}n)$ *time and polynomial space.*

3.3. Algorithm on trees

In this subsection, we show an $O((b+1)^k n)$-time algorithm for the Saving All But k-Vertices problem on trees.

Theorem 3.7. Saving All But k-Vertices *on trees is solvable in time* $O((b+1)^k n)$ *and polynomial space.*

Proof. If the root s has at most b children, then we immediately answer Yes. We may assume that the root has exactly $a \geqslant b+1$ children, and $k \geqslant a-b$ since otherwise we simply answer No. We use Lemma 2.11 and branch on every subset of b children of the root s. In each branch, we cut the subtree rooted at the protected vertex, identify all the vertices that are on fire after the first round, and decrease the parameter by $a-b$. In this way, we obtain a new instance of the Saving All But k-Vertices problem with parameter value equal to $k-(a-b)$. The time bound follows from the inequality

$$T(k) \leqslant \binom{a}{b} T\big(k-(a-b)\big) + O(n)$$

which is worst when $a = b+1$. □

3.4. Kernelization feasability

In this section, we provide an $O(2^\tau k\tau)$ kernel for Saving All But k-Vertices.

Theorem 3.8. Saving All But k-Vertices *admits a kernel of size at most* $O(2^\tau k\tau)$.

Proof. First, we compute in polynomial-time a 2-approximate vertex cover $C' \subseteq V$ where $|C'| = \tau' \leqslant 2\tau$. We may assume that $b < \tau'$, since otherwise the answer is Yes. Indeed, suppose to the contrary that $b \geqslant \tau'$. We may assume that $|N(s)| < b+k$, because otherwise the answer is clearly No. If $s \notin C'$, then it is enough to protect all the vertices in C' at the first round to stop the fire. If $s \in C'$, then let $N_1(s) = N(s) \cap C'$ and $N_2(s) = N(s) \setminus C'$. If $|N_2(s)| < k$, then the strategy that protects all the vertices in C' at the first round burns at most k vertices. If $|N_2(s)| \geqslant k$, then consider the following strategy. At the first time step, protect all the vertices in $N_1(s)$, and $b - |N_1(s)|$ vertices of $N_2(s)$ (notice that $|N_1(s)| < b$). At the second time step, protect all the vertices in $C' \setminus N_1[s]$. Hence, the number of burned vertices using this strategy is at most k.

The reduction is the same as the one describes in Theorem 2.20, but we use the following slightly different reduction rule.

Rule: Let $T \subseteq V$ be a twins set. If $|T| \geqslant kb$, then delete $|T| - kb$ vertices of T.

Using similar arguments as in Theorem 2.20, we get a kernel of size $O(2^\tau kb) = O(2^\tau k\tau') = O(2^\tau k\tau)$. □

3.5. No polynomial-size kernel for trees

In this section, we show that both Saving All But k-Vertices and k-Vertex Protection have no polynomial kernel for budget $b = 1$, unless NP $\subseteq$ coNP/poly. Next, we provide a generalization of these results to any fixed budget $b \geqslant 2$.

Theorem 3.9. *Unless* NP $\subseteq$ coNP/poly, *there is no polynomial kernel for the* Saving All But k-Vertices *problem, even if the input graph is a tree of maximum degree four.*

Before we prove Theorem 3.9 we describe the necessary tools. We use the cross-composition technique introduced by Bodlaender et al. [6], which is based on the previous results of Bodlaender et al. [5] and Fortnow and Santhanam [17]. We recall the crucial definitions.

Definition 3.10 *(Polynomial equivalence relation).* (See [6].) An equivalence relation $\mathcal{R}$ on Σ^* is called a *polynomial equivalence relation* if (1) there is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether $\mathcal{R}(x,y)$ in $(|x|+|y|)^{O(1)}$ time; (2) for any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into at most $(\max_{x\in S}|x|)^{O(1)}$ classes.

Definition 3.11 *(Cross-composition).* (See [6].) Let $L \subseteq \Sigma^*$, and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L *cross-composes* into Q if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which, given t strings $x_1, x_2, \ldots x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^t |x_i|$ such that (1) $(x^*, k^*) \in Q$ iff $x_i \in L$ for some $1 \leqslant i \leqslant t$; (2) k^* is bounded polynomially in $\max_{i=1}^t |x_i| + \log t$.

Theorem 3.12. *(See [6, Theorem 9].) If $L \subseteq \Sigma^*$ is NP-hard under Karp reductions and L cross-composes into the parameterized problem Q that has a polynomial kernel, then* NP $\subseteq$ coNP/poly.

We apply Theorem 3.12, where as the language L we use Saving All But k-Vertices in trees of maximum degree three, which is NP-complete [14]. To finish the proof of Theorem 3.9, we present a cross-composition algorithm.

Lemma 3.13. *The unparameterized version of the* Saving All But k-Vertices *problem on trees with maximum degree three cross-composes to* Saving All But k-Vertices *on trees with maximum degree four.*

Proof. Observe that any polynomial equivalence relation is defined on all words over the alphabet Σ and for this reason we should also define how the relation behaves on words that do not represent instances of the problem. For the equivalence relation $\mathcal{R}$ we take a relation that puts all malformed instances into one equivalence class and all well-formed instances are grouped according to the number of vertices we are allowed to burn.

If we are given malformed instances, we simply output a trivial No-instance. Thus in the rest of the proof we assume we are given a sequence of instances $(T_i, s_i, k)_{i=1}^{t}$ of the unparameterized version of Saving All But k-Vertices, where each T_i is of maximum degree three. Observe that in all instances we have the same value of the parameter k. W.l.o.g. we assume that $t = 2^h$ for some integer $h \geqslant 1$. Otherwise we can duplicate an appropriate number of instances (T_i, s_i, k).

We create a new tree T' as follows. Let T' be a full binary tree with exactly t leaves rooted at a vertex s'. Now for each $i = 1, \ldots, t$, we replace the i-th leaf of the tree by tree T_i rooted at s_i. Finally, we set $k' = k + h = k + \log_2 t$. Observe that since each tree T_i is of maximum degree three, the tree T' is of maximum degree four. To prove correctness, it is enough to show that any strategy that minimizes the number of burned vertices protects exactly one vertex at each depth $1, \ldots, h$, which follows from Lemma 2.11. Hence in any strategy that minimizes the number of burned vertices, there will be exactly one vertex s_i which is on fire after h rounds. □

We can obtain a similar result for the decision variant of Maximum k-Vertex Protection.

Theorem 3.14. *Unless* NP $\subseteq$ coNP/poly, *there is no polynomial kernel for the* k-Vertex Protection *problem, even if the input graph is a tree of maximum degree four.*

Proof. There are only two differences compared to the proof for Saving All But k-Vertices.

- For the equivalence relation $\mathcal{R}$, we take a relation that puts all malformed instances into one equivalence class, and all well-formed instances are grouped according to the number of vertices of the tree, the parameter value k, and the value a.
- The value of k' for the tree T' is $k + h$, and the value of a' is equal to $a + (t-1)n + (t-h-1)$, where n is the number of vertices in each of the trees T_i. The additional summands are derived from the fact that any optimal strategy will ensure that after h rounds exactly one vertex s_i will be on fire and hence we save $t-1$ subtrees rooted at s_i, each containing n vertices, and $t-h-1$ vertices of the full binary tree.

This completes the proof. □

We may generalize the previous results using the fact that the unparameterized version of Saving All But k-Vertices is NP-complete for trees of maximum degree $b+3$ where $b \geqslant 2$ is fixed [3].

Theorem 3.15. *For any fixed budget $b \geqslant 2$, there is no polynomial kernel for the* Saving All But k-Vertices *problem, even if the input graph is a tree of maximum degree $b+4$, unless* NP $\subseteq$ coNP/poly.

Proof. Let $b \geqslant 2$ be any fixed constant. We use Theorem 3.12, where L is the unparameterized version of Saving All But k-Vertices on trees of maximum degree $b+3$. To give the cross-composition, we use the same proof from Lemma 3.13 with the following differences.

- For the equivalence relation $\mathcal{R}$, we take a relation that puts all malformed instances into one equivalence class, and all well-formed instances are grouped according to the parameter value k and the budget b.
- We assume that $t = (b+1)^h$ for some integer $h \geqslant 1$.
- The tree T' is now a full $(b+1)$-ary tree with exactly t leaves, rooted at a vertex s' — that is, a tree in which every vertex other than the leaves has $b+1$ children and all leaves are at the same distance from the root. Finally, we set $k' = k + \log_{b+1} t$.

This completes the proof. □

Theorem 3.16. *For any fixed budget $b \geqslant 2$, there is no polynomial kernel for the k-Vertex Protection problem, even if the input graph is a tree of maximum degree $b + 4$, unless* NP $\subseteq$ coNP/poly.

Proof. Using the construction from Theorem 3.15 and modifications from Theorem 3.14, the result follows. □

4. Open problems

In this paper, we refined and extended several parameterized algorithmic and complexity results about different parameterizations of the b-Firefighter problem. We conclude with the following open problems.

- We have shown that Saving k-Vertices is in FPT on graphs of bounded local treewidth, and thus on planar graphs, by showing that the problem is in FPT parameterized by k and the treewidth of a graph. While Maximum k-Vertex Protection is also in FPT parameterized by k and the treewidth, we do not know if the problem is in FPT on planar graphs, and leave it as an open problem.
- Furthermore, we do not know if Saving All But k-Vertices is fixed-parameter tractable when parameterized by tw and k. One could try to adapt the MSOL formula for Maximum k-Vertex Protection, but in its current form the size of the formula depends on k and b. This is not a problem for Maximum k-Vertex Protection, as we can assume that $b \leqslant k$. However, it is not clear whether a similar assumption can be made for Saving All But k-Vertices.
- The b-Firefighter problem is solvable in subexponential time on trees. Is it solvable in time $2^{o(n)}$ on n-vertex planar graphs? Even the case of outerplanar graphs is open.
- It is unknown if Saving k-Vertices, Saving All But k-Vertices, and k-Vertex Protection admit a polynomial kernel for parameters τ and k. While we have shown that k-Vertex Protection and Saving All But k-Vertices do not admit a polynomial kernel for parameters k and tw, we do not know if it also the case for Saving k-Vertices.
- We have proved that Saving k-Vertices and Maximum k-Vertex Protection can be solved in time $n^{O(k)}$. We also showed that there is no $n^{o(\sqrt{k})}$ algorithm for these problems unless FPT = M[1]. However, the existence of an $n^{o(k)}$-time algorithm is an open problem.
- Finally, we do not know if any of the three parameterized versions of the problem is solvable in parameterized subexponential time $2^{o(k)}n^{O(1)}$ on trees.

Acknowledgments

We thank Leizhen Cai for pointing us to [20] and for sending us the full version of [8]. We also acknowledge the support of Schloss Dagstuhl for Seminar 11071 (GRASTA 2011 — Theory and Applications of Graph Searching Problems). Research of Fedor Fomin was supported by the European Research Council (ERC) grant "Rigorous Theory of Preprocessing", reference 267959. Research of Michael R. Fellows was supported by the Australasian Research Council, grant number DP1097129. Research of Marek Cygan was partially supported by grant no. N206 567140 of the National Science Centre and Foundation For Polish Science. The preliminary results of this publication were presented at IPEC2011 [10] and ISAAC2011 [2].

References

[1] S. Arnborg, J. Lagergren, D. Seese, Easy problems for tree-decomposable graphs, J. Algorithms 12 (1991) 308–340.
[2] C. Bazgan, M. Chopin, M. Fellows, Parameterized complexity of the firefighter problem, in: ISAAC, in: Lect. Notes Comput. Sci., vol. 7074, 2011, pp. 643–652.
[3] C. Bazgan, M. Chopin, B. Ries, The firefighter problem with more than one firefighter on trees, Discrete Appl. Math. 161 (7–8) (2013) 899–908.
[4] H.L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (6) (1996) 1305–1317.
[5] H.L. Bodlaender, R.G. Downey, M.R. Fellows, D. Hermelin, On problems without polynomial kernels, J. Comput. Syst. Sci. 75 (8) (2009) 423–434.
[6] H.L. Bodlaender, B.M.P. Jansen, S. Kratsch, Cross-composition: A new technique for kernelization lower bounds, in: STACS, in: LIPIcs, vol. 9, 2011, pp. 165–176.
[7] L. Cai, Parameterized complexity of cardinality constrained optimization problems, Comput. J. 51 (1) (2008) 102–121.
[8] L. Cai, E. Verbin, L. Yang, Firefighting on trees: $(1 - 1/e)$-approximation, fixed parameter tractability and a subexponential algorithm, in: ISAAC, in: Lect. Notes Comput. Sci., vol. 5369, 2008, pp. 258–269.
[9] J. Chen, X. Huang, I.A. Kanj, G. Xia, Strong computational lower bounds via parameterized complexity, J. Comput. Syst. Sci. 72 (8) (2006) 1346–1367.
[10] M. Cygan, F. Fomin, E.J. van Leeuwen, Parameterized complexity of firefighting revisited, in: IPEC, in: Lect. Notes Comput. Sci., vol. 7112, 2011, pp. 13–26.
[11] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer-Verlag, New York, 1999.
[12] D. Eppstein, Diameter and treewidth in minor-closed graph families, Algorithmica 27 (3) (2000) 275–291.
[13] S. Finbow, B. Hartnell, Q. Li, K. Schmeisser, On minimizing the effects of fire or a virus on a network, J. Comb. Math. Comb. Comput. 33 (2000) 311–322.
[14] S. Finbow, A. King, G. MacGillivray, R. Rizzi, The firefighter problem for graphs of maximum degree three, Discrete Math. 307 (16) (2007) 2094–2105.
[15] S. Finbow, G. MacGillivray, The firefighter problem: a survey of results, directions and questions, Australas. J. Comb. 43 (2009) 57–77.
[16] P. Floderus, A. Lingas, M. Persson, Towards more efficient infection and fire fighting, Int. J. Found. Comput. Sci. 24 (1) (2013) 3–14.
[17] L. Fortnow, R. Santhanam, Infeasibility of instance compression and succinct PCPs for NP, in: STOC 2008: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, ACM, 2008, pp. 133–142.
[18] G. MacGillivray, P. Wang, On the firefighter problem, J. Comb. Math. Comb. Comput. 47 (2003) 83–96.
[19] L. Mathieson, S. Szeider, Editing graphs to satisfy degree constraints: A parameterized approach, J. Comput. Syst. Sci. 78 (1) (2012) 179–191.
[20] L. Yang, Efficient Algorithms on Trees, M. Phil. thesis, Department of Computer Science and Engineering, The Chinese University of Hong Kong, 2009.