ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/221543154

Politician's Firefighting

Conference Paper · December 2006

DOI: 10.1007/11940128_61 · Source: DBLP

CITATIONS
5

READS
77

3 authors, including:

Ulrike Stege
University of Victoria
88 PUBLICATIONS **1,299** CITATIONS
SEE PROFILE

Norbert Zeh
Dalhousie University
109 PUBLICATIONS **1,113** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Indexing very large strings View project

All content following this page was uploaded by Ulrike Stege on 30 May 2014.

The user has requested enhancement of the downloaded file.

Politician's Firefighting

Allan E. Scott[1], Ulrike Stege[1], and Norbert Zeh[2,⋆]

[1] Department of Computer Science, University of Victoria, Victoria, Canada
{aescott,stege}@cs.uvic.ca
[2] Faculty of Computer Science, Dalhousie University, Halifax, Canada
nzeh@cs.dal.ca

Abstract. Firefighting is a combinatorial optimization problem on graphs that models the problem of determining the optimal strategy to contain a fire and save as much (trees, houses, etc.) from the fire as possible. We study a new version of firefighting, which we call *Politician's Firefighting* and which exhibits more locality than the classical one-firefighter version. We prove that this locality allows us to develop an $O(bn)$-time algorithm for this problem on trees, where b is the number of nodes initially on fire. We further prove that Politician's Firefighting is NP-hard on planar graphs of degree at least 5, and we present an $O(m + k^{2.5}4^k)$-time algorithm for this problem on general graphs, where k is the number of nodes that burn using the optimal strategy, thereby proving that it is fixed-parameter tractable. We present experimental results that show that our algorithm's search tree size is in practice much smaller than the worst-case bound of 4^k.

1 Introduction

Firefighting can be thought of as a puzzle game where the player's goal is to save nodes in a graph from an advancing fire. Given a graph $G = (V, E)$ and a set $B_0 \subset V$ of initially burning nodes, the game proceeds in rounds, numbered 0 through r. In each round, first the fire advances and then the player places firefighters on one or more nodes that are neither burning nor occupied (by a firefighter). Once a node is burning or occupied, it stays that way for the rest of the game. Round 0 is special; all nodes in B_0 are set on fire. In subsequent rounds, the fire spreads from each burning node to every adjacent unoccupied node. The game ends when the fire can no longer spread, that is, when all neighbours of burning nodes are burning or occupied. Viewed as an optimization problem, the player's goal is to find a firefighter-placement strategy that minimizes the number of burning nodes at the end of the game. The problem can be seen as a model of the spread of forest fires or diseases in social networks, which motivated the initial study of this problem [3, 4, 7].

The classical version of firefighting, introduced in [4] allows the player to place *one* firefighter on an arbitrary node in each round, so long as the node is

⋆ Research supported by the Natural Sciences and Engineering Research Council of Canada and the Canadian Foundation for Innovation.

not yet occupied or burning. Finding an optimal strategy under these conditions is NP-hard even on trees of degree three and with $|B_0| = 1$, provided that the node in B_0 has degree three [1]. It is not hard to see that a simple greedy algorithm produces an optimal strategy on binary trees. On arbitrary trees the greedy strategy that always protects the heaviest threatened subtree produces a 2-approximation w.r.t. the number of saved nodes [5]. In [8], algorithms for containing the fire on 2- and 3-dimensional grids are studied. Apart from the results cited here, we are not aware of any other algorithmic results for this problem. What makes this version of firefighting hard is the complete freedom where to place the firefighter, that is, the non-local nature of the problem.

In this paper, we study *Politician's Firefighting*, a more localized version of the problem: In each round, we are allowed to deploy as many firefighters as there are burning nodes (politicians allocate resources according to how dire the situation is, that is, how many nodes are burning). However, if a node x "generates" a firefighter by being on fire, this firefighter can only be placed on an unoccupied, non-burning neighbour of x. In other words, we can actually use only as many firefighters as there are burning nodes with unoccupied non-burning neighbours.

We feel that is is more realistic to allow more than one firefighter to be placed in each round because typically more than one fire brigade fights a forest fire. The constraints imposed on where firefighters may be placed reflect the political reality that politicians and local inhabitants would prefer to see their fire brigade protect them or their neighbours, rather than somebody miles away. This constraint can also be seen as a logistic one since fire trucks travel at a finite speed. Another motivation for the locality is that, when using vaccination to contain the spread of a disease, one usually vaccinates persons interacting with infected persons before using a much wider "radius" of vaccination, particularly if vaccine is expensive or hard to obtain.

We prove in Section 2 that this problem can be solved in $O(bn)$ time on trees, where $b = |B_0|$. In Section 3, we show that Politician's Firefighting is NP-hard even on planar graphs of degree 5. In Section 4, we present an $O(m+k^{2.5}4^k)$-time algorithm for general graphs, which shows that the problem is fixed-parameter tractable. The worst-case bound on the size of the search tree in our algorithm is tight. However, experimental results discussed in Section 4 indicate that, in practice, the search-tree size is much smaller.

2 Trees

We start by arguing that the locality of Politician's Firefighting helps to solve it in polynomial time on trees. We choose an arbitrary node in B_0 as the root of the tree. For every node v, let T_v be the subtree rooted in v, let p_v be v's parent, and let C_v be the set of v's children. Our strategy is to consider all possible cases how a node v may be set on fire or saved and, based on their analysis, develop a recurrence for the number of nodes that burn in T_v in each case. This allows us to use dynamic programming to determine the number of nodes that burn

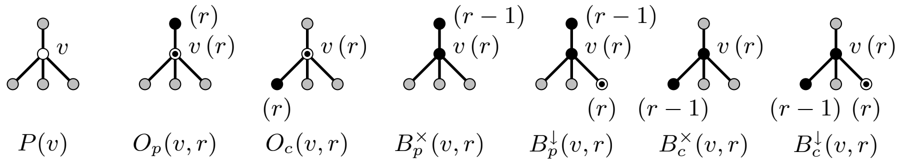

Fig. 1. The different states of a node v and the possible choices how v can attain this state. Burning nodes are black, occupied nodes are white with a black dot inside, and unoccupied non-burning nodes are white. The state of a gray node is unspecified. Labels in parentheses show when the nodes attain the shown state.

using the optimal strategy. (The corresponding strategy is then easily found.) A straightforward evaluation of these recurrences leads to a running time of $\Theta(n^4)$ in the worst case. We then discuss how to reduce the running time to $O(bn)$.

2.1 The Basic Algorithm

Consider a node v. At the end of the game, v is in one of three states: *burning*, *occupied* by a firefighter, or *protected*; the latter means that there is no firefighter on v, but there is a firefighter on every path between v and a burning node. A burning node v either belongs to B_0 or is set on fire by one of its neighbours in some round r. We distinguish whether it is p_v or a child of v that sets v on fire. We further distinguish whether or not we choose to place v's firefighter on a child of v in round $r+1$. Similarly, an occupied node has a firefighter placed on it in some round r. This firefighter is available because p_v or a child of v catches fire in round r. We use the following notation to denote the number of nodes that burn in T_v in each of the resulting cases (see Figure 1 for an illustration): $P(v)$ if v is protected; $O_p(v,r)$ and $O_c(v,r)$ if v is occupied in round r; $B_c^{\times}(v,r)$, $B_c^{\downarrow}(v,r)$, $B_p^{\times}(v,r)$, and $B_p^{\downarrow}(v,r)$ if v is set on fire in round r. Subscripts p and c denote the subcases when v is set on fire by its parent or a child, respectively, or when the parent's or a child's firefighter is placed on v. Superscripts indicate whether we place v's firefighter on one of v's children ($\downarrow$) or not ($\times$). In addition, we use the following notation: $B_p(v,r) = \min(B_p^{\downarrow}(v,r), B_p^{\times}(v,r))$, $B_c(v,r) = \min(B_c^{\downarrow}(v,r), B_c^{\times}(v,r))$, $O_c^*(v,r) = \min_{1\le r'\le r} O_c(v,r')$, $B_c^*(v,r) = \min_{r\le r'\le n} B_c(v,r')$, and $L(v,r) = \min(B_c(v,r-1), B_c(v,r), B_c(v,r+1), O_c^*(v,r), B_p(v,r+1))$.

Since $|B_0| = b$, every node v in T can be occupied or set on fire only in b different rounds, corresponding to the lengths of the paths from v to the nodes in B_0. If node v cannot be set on fire or occupied at time r, we define its corresponding $B_{\cdot}^{\cdot}(v,r)$ or $O_{\cdot}(v,r)$ value to be $+\infty$.

Next we derive a recurrence for $B_c^{\downarrow}(v,r)$. Due to lack of space, we only state the recurrences for the other cases; they are easily obtained using similar, but simpler, analyses. For technical reasons, we treat every node $v \in B_0$ as being set on fire in round 0 by both an imaginary child and an imaginary parent. Thus, for $v \in B_0$ and $r > 0$, we have $B_c^{\downarrow}(v,r) = +\infty$; for $r = 0$, we have $B_c^{\downarrow}(v,0) = 1 + \min_{w\in C_v}\bigl(O_p(w,0) + \sum_{w'\in C_v\setminus\{w\}} L(w',0)\bigr)$ because, after being set on fire, node v chooses one child w to occupy in round 0; any other child $w' \in C_v \setminus \{w\}$ is then either set on fire by v in round 1, set on fire by a child in round 0 or 1, or occupied using one of its children's firefighters in round 0.

For $v \notin B_0$, node v is set on fire by one of its children, w_1, and again node v chooses a child $w_2 \in C_v \setminus \{w_1\}$ to occupy using v's firefighter; any other node $w' \in C_v \setminus \{w_1, w_2\}$ is set on fire by v in round $r+1$, set on fire by one of its own children in round $r-1$, r, or $r+1$, or occupied by one of its children's firefighters no later than round r. This leads to the following recurrence:

$$B_c^{\downarrow}(v,r) = 1 + \min_{\substack{w_1,w_2 \in C_v \\ w_1 \neq w_2}} \left(B_c(w_1, r-1) + O_p(w_2, r) + \sum_{w' \in C_v \setminus \{w_1,w_2\}} L(w', r) \right)$$

Using similar analyses, we obtain for $v \in B_0$: $O_c(v,r) = O_p(v,r) = P(v) = +\infty$. For $r > 0$, we have $B_c^{\times}(v,r) = B_p^{\downarrow}(v,r) = B_p^{\times}(v,r) = +\infty$. For $r = 0$, we have $B_p^{\downarrow}(v,0) = B_c^{\downarrow}(v,0)$ and $B_c^{\times}(v,0) = B_p^{\times}(v,0) = 1 + \sum_{w \in C_v} L(w,0)$ For $v \notin B_0$, we obtain

$$\begin{aligned}
B_c^{\times}(v,r) &= 1 + \min_{w \in C_v} \left(B_c(w, r-1) + \sum_{w' \in C_v \setminus \{w\}} L(w', r) \right) \\
B_p^{\downarrow}(v,r) &= 1 + \min_{w \in C_v} \left(O_p(w, r) + \sum_{w' \in C_v \setminus \{w\}} L(w', r) \right) \\
B_p^{\times}(v,r) &= 1 + \sum_{w \in C_v} L(w, r) \\
O_c(v,r) &= \min_{w \in C_v} \left(B_c^{\times}(w,r) + \sum_{w' \in C_v \setminus \{w\}} \min(P(w'), B_c^*(w',r), O_c^*(w',n)) \right) \\
O_p(v,r) &= \sum_{w \in C_v} \min(P(w), B_c^*(w,r), O_c^*(w,n)) \\
P(v) &= \sum_{w \in C_v} \min(O_c^*(w,n), P(w))
\end{aligned}$$

Each of these recurrences for a given node v depends only on values of the recurrences on children of v. Hence, they can be computed bottom-up. Since the game ends after at most n rounds, we must consider up to n different time values. The most expensive recurrence to evaluate is $B_c^{\downarrow}(v,r)$, where we must consider all pairs of children (w_1, w_2) of v. The number of these pairs, summed over all nodes in T, is $\Theta(n^2)$ in the worst case. For each pair, we spend linear time to evaluate the expression inside the outer parentheses, leading to a $\Theta(n^3)$ bound per round. Summing over all n rounds gives a running time of $\Theta(n^4)$.

2.2 A Faster Algorithm

To reduce the running time to $O(n^2)$, we need to evaluate every recurrence for a given pair (v,r) in $O(1 + |C_v|)$ time. This is easy for $P(v)$, $O_p(v,r)$, and

$B_p^\times(v,r)$. Next we discuss how to achieve this bound for evaluating $O_c(v,r)$, $B_c^\downarrow(v,r)$, $B_c^\times(v,r)$, and $B_p^\downarrow(v,r)$. We discuss $B_c^\downarrow(v,r)$ in detail; the same ideas also speed up the computation of the other recurrences. If we precompute the sum $L^*(v,r) = \sum_{w \in C_v} L(w,r)$, which takes $O(|C_v|)$ time, we can rewrite the recurrence for $B_c^\downarrow(v,r)$ as

$$B_c^\downarrow(v,r) = 1 + \min_{\substack{w_1,w_2 \in C_v \\ w_1 \neq w_2}} \bigl(L^*(v,r) + B_c(w_1, r-1) - L(w_1,r) + O_p(w_2,r) - L(w_2,r)\bigr),$$

which can be evaluated in $O(1 + |C_v|^2)$ time. Looking more closely at the rewritten form of $B_c^\downarrow(v,r)$, we observe that $B_c^\downarrow(v,r)$ is minimized if $B'(w_1,r) = B_c(w_1,r-1) - L(w_1,r)$ and $B''(w_2,r) = O_p(w_2,r) - L(w_2,r)$ are minimized, except that w_1 and w_2 cannot be the same node. Thus, if w_1' and w_1'' are the two children of v that minimize $B'(w,r)$ and w_2' and w_2'' are the two children of v that minimize $B''(w,r)$, we have three cases: Assume w.l.o.g. that $B'(w_1',r) \leq B'(w_1'',r)$ and $B''(w_2',r) \leq B''(w_2'',r)$. If $w_1' \neq w_2'$, let $w_1 = w_1'$ and $w_2 = w_2'$. If $w_1' = w_2'$ and $B(w_1'',r) - B(w_1',r) \leq B(w_2'',r) - B(w_2',r)$, then let $w_1 = w_1''$ and $w_2 = w_2'$; otherwise, let $w_1 = w_1'$ and $w_2 = w_2''$.

Nodes w_1', w_1'', w_2', w_2'' can be found in $O(|C_v|)$ time. Once this is done, $B_c^\downarrow(v,r)$ can be evaluated in constant time because one of the three combinations (w_1', w_2'), (w_1', w_2''), (w_1'', w_2') minimizes $B_c^\downarrow(v,r)$. Hence, each recurrence can be evaluated in $O(1 + |C_v|)$ time per pair (v,r), and the total cost of evaluating the recurrences over all nodes is $O(n)$ per time value r. Since we have to consider only $1 \leq r \leq n$, the total running time is $O(n^2)$.

To reduce the running time to $O(bn)$, we observe that every node can be set on fire or occupied by a neighbour at only b different times, determined by the distances from v to the nodes in B_0. Thus, we must evaluate each recurrence for only b different time values for each node in T; we define every value that is not computed explicitly to be $+\infty$. This reduces the running time to $O(bn)$.

Theorem 1. *Politician's Firefighting can be solved in $O(bn)$ time on a tree with n nodes of which b are initially on fire.*

3 NP-Hardness on Planar Graphs

In this section, we prove that Politician's Firefighting is NP-hard:

Theorem 2. *Politician's Firefighting is NP-hard, even on planar graphs with vertices of degree at most five and only one node initially on fire.*

We prove NP-hardness of Politician's Firefighting by reduction from Planar Vertex Cover [2]. In particular, given a planar graph G, we construct another planar graph G' with nodes[3] of degree at most 5, and a set $B_0 = \{\rho\}$, where ρ

[3] To avoid confusion, we refer to the vertices of G as "vertices" and to the vertices of G' as "nodes".

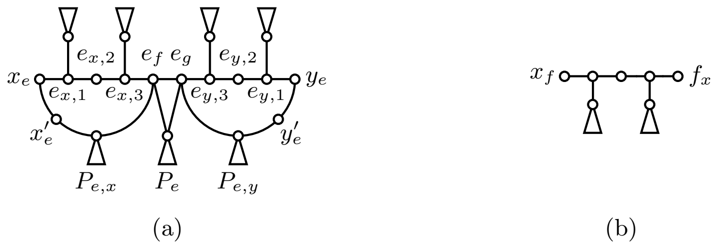

Fig. 2. (a) The edge widget $\mathcal{E}_e$ for the edge $e = xy$. (b) A connector.

is an almost arbitrary node of G', such that G has a vertex cover of size k if and only if G' has a firefighting strategy that burns only "a few" nodes.

The construction of G' has the following intuition: First we replace the vertices and edges of G with subgraphs called *vertex widgets* and *edge widgets*. A vertex widget is built such that if any one of its nodes burn, n^3 nodes burn in the widget. An edge widget is built such that letting one or both of two special *incineration nodes* burn sacrifices n^5 nodes, unless we let at least one of the vertex widgets corresponding to the endpoints of the edge burn as well. We complete the construction by superimposing two additional graph structures on the vertex and edge widgets. The first one allows the fire to spread from ρ to all incineration nodes, and is built so that we cannot prevent the spread unless we sacrifice n^5 nodes elsewhere. Thus, if the size of a minimum vertex cover of G is k, at least n^5 nodes in G' will burn unless we let k vertex widgets in G' burn, which means that roughly kn^3 nodes burn. The second graph structure allows the fire to spread to the k vertex widgets corresponding to vertices in a vertex cover, without using nodes in edge widgets.

We use *penalizers* to ensure that letting certain nodes in G' burn causes many more nodes to burn. These are complete ternary trees whose leaves are at depth d, for some $d > 0$. When the root of a penalizer P catches fire, the optimal strategy burns a complete binary subtree of P of height d. Thus, we have

Lemma 1. *If the root of a penalizer P of height d catches fire, the optimal strategy burns $2^{d+1} - 1$ nodes in P.*

We call a penalizer *small* if $2^{d+1} - 1 = n^3$, and *big* if $2^{d+1} - 1 = n^5$. Both are of polynomial size: small penalizers have size $O(n^{3 \log 3})$, big penalizers have size $O(n^{5 \log 3})$. We say a node v of G' is *adjacent to a penalizer P* if v is adjacent to the root of P and no other node in P is adjacent to a node in $G' - P$.

Next we define the different widgets that comprise G' and discuss how they are connected. For the construction, we assume that we are given a planar embedding of G. From the construction, it will be obvious that G' is planar and that every node in G' has degree at most five.

Vertex widgets. Let x be a vertex of G, and let $e_1, f_1, \ldots, e_d, f_d$ be the edges and faces incident to x, in clockwise order. The vertex widget $\mathcal{V}_x$ consists of a simple cycle $(x_{e_1}, x_{f_1}, \ldots, x_{e_d}, x_{f_d})$. Each node of this cycle is adjacent to a small penalizer, except x_{e_1}, which is adjacent to two small penalizers. Note that, once a single node of the cycle burns we have two choices: let the fire spread around the cycle and protect one penalizer per node, or protect a cycle-neighbour of a burning node. In the former case, we let the second penalizer attached to x_{e_1} burn, incurring a penalty of n^3 burning nodes. In the latter case, the penalizers attached to the node whose cycle-neighbour we protect burn. This incurs a penalty of at least n^3 burning nodes.

Edge widgets. Let e be an edge with endpoints x and y and incident faces f and g. In G', edge e is represented by an edge widget $\mathcal{E}_e$, shown in Figure 2a. The endpoints x_e and y_e are shared between the edge widget $\mathcal{E}_e$ and the vertex widgets $\mathcal{V}_x$ and $\mathcal{V}_y$; that is, the endpoints of $\mathcal{E}_e$ are the same nodes as the nodes with the same names in $\mathcal{V}_x$ and $\mathcal{V}_y$. All penalizers in the edge widget are big. We argue later that we have to let both e_f and e_g burn. We call e_f and e_g *incineration nodes*, as we cannot protect all three penalizers threatened by these two nodes unless we let at least one of the nodes x'_e and y'_e burn, which can be achieved only by letting $\mathcal{V}_x$ or $\mathcal{V}_y$ (or both) burn.

Face widgets. G' contains one face widget $\mathcal{F}_f$ per face f of G. Similar to a vertex widget, the face widget for a face f with incident vertices and edges $x_1, e_1, \ldots, x_d, e_d$, in this order clockwise around f, consists of a cycle $(f_{x_1}, f_{e_1}, \ldots, f_{x_d}, f_{e_d})$, each of whose nodes has an attached penalizer; but this time the penalizers are big. Once one node in $\mathcal{F}_f$ catches fire, the only way we can prevent n^5 nodes from burning is to let the fire spread around $\mathcal{F}_f$, while protecting the roots of all penalizers in $\mathcal{F}_f$.

The last two widgets build two additional graph structures within G'. The first allows us to cheaply set fire to vertex widgets corresponding to vertices in a vertex cover of G. The second ensures that every node e_f or e_g in an edge widget burns eventually, forcing us to set fire to at least one vertex widget incident to each edge widget in order to save all penalizers in the edge widget.

Channels. A channel is a path of length $42n$. Each of its internal nodes has a big penalizer attached to it. The first endpoint of the channel belongs to a face widget; the second endpoint belongs to an edge widget. More precisely, for every face f and every edge e on its boundary, there is a channel in G' whose endpoints are f_e and e_f. Note that this implies that, once one of the endpoints of the channel burns, we have to protect its big penalizer inside the face or edge widget. This sets fire to its neighbour inside the channel. In order to prevent n^5 nodes from burning in the channel, we now have to let the fire spread along the channel path and, for every node on the path, protect its adjacent penalizer.

Connectors. A connector is used to let the fire spread cheaply between face and vertex widgets. For every face f and every vertex x on its boundary, there is a connector with endpoints x_f and f_x, which belong to $\mathcal{V}_x$ and $\mathcal{F}_f$; see Figure 2b. For each connector, if f_x burns, we have two choices: Either we let the fire spread along the connector, thereby forcing all cycle nodes in

$\mathcal{V}_x$ to burn, or we let the fire spread to the middle node and then stop the fire by placing this node's firefighter on the neighbour of x_f in the connector.

To finish the construction of the firefighting instance, we choose an arbitrary non-penalizer node ρ in a face widget and define $B_0 = \{\rho\}$. The following lemma proves that connectors and face widgets allow us to cheaply and quickly set fire to the appropriate vertex widgets in G'. Lemma 3 then uses this fact to prove that G has a small vertex cover if and only if G' has a firefighting strategy that lets few nodes burn.

Lemma 2. *Let $V' = \{x_1, \ldots, x_k\}$ be a vertex cover of G. Then G' contains a connected subgraph containing only nodes from face widgets, connector widgets, and vertex widgets $C_{x_1}, \ldots, C_{x_k}$. This graph includes ρ and has diameter at most $42n - 84$.*

Lemma 3. *Graph G has a vertex cover of size k if and only if G' has a strategy that burns at most $kn^3 + 252n^2 - 432n - 144$ nodes.*

Proof sketch. We prove only the "only if" part. The "if" part can be proved using similar arguments. Let $V' = \{x_1, \ldots, x_k\}$ be a vertex cover of size k, and let H be a subgraph of G' as in Lemma 2. Then, due to the diameter bound of H, we can make sure that every node in H burns by time $42n - 84$. Moreover, we can protect all penalizers incident to nodes in H, except one small penalizer per vertex widget. Thus, we let kn^3 nodes in penalizers burn.

Now observe that an incineration node in an edge widget $\mathcal{E}_e$ for an edge $e = xy$ catches fire no earlier than round $42n$, unless we let the fire spread to this node from x_e or y_e. In summary, for every edge widget $\mathcal{E}_e$, at least one of x_e and y_e catches fire at least 84 rounds before e_f or e_g can be set on fire through a channel. We sketch here what happens if only x_e burns, that is, $y \notin V'$. The other two cases are similar.

In this case, we let the fire spread from x_e to x'_e and $e_{x,1}$, using x_e's firefighter to protect its incident penalizer inside $\mathcal{V}_x$. In the next time step, we use the firefighters of x'_e and $e_{x,1}$ to protect the adjacent penalizers, letting $e_{x,2}$ burn. Next we let the fire spread from $e_{x,2}$ to $e_{x,3}$ and then on to e_f, e_g, $e_{y,3}$, and $e_{y,2}$. Each of these nodes has only one unprotected adjacent penalizer by the time it catches fire because x'_e has already protected $P_{e,x}$ and then e_f protects P_e before e_g catches fire. Thus, each node can use its firefighter to protect the one penalizer it threatens. Finally, we use $e_{y,2}$'s firefighter to protect $e_{y,1}$, thereby preventing the fire from spreading into $\mathcal{V}_y$.

To obtain the claimed bound on the total number of nodes that burn, we count the total number of non-penalizer nodes in vertex widgets, face widgets, channels, and connectors, and add the kn^3 nodes that burn in small penalizers in the k vertex widgets $\mathcal{V}_{x_1}, \ldots, \mathcal{V}_{x_k}$. □

4 Fixed-Parameter Tractability on General Graphs

We present a bounded search-tree algorithm that solves Politician's Firefighting in $O(m + k^{2.5}4^k)$ time. Rather than deciding an entire round at once, our algorithm is based on the idea of choosing a single *threatened node* v (i.e., a non-burning node adjacent to a burning node), and branching recursively on two cases: place a firefighter on v, or let v burn. However, there are two problems that must be addressed for the algorithm to work.

The first problem arises because we decouple the recursion from the rounds. Specifically, we have to track the set of nodes threatened from the beginning of the round since we place fires during the round rather than at the beginning of the next round. Otherwise, new nodes would become threatened during the round as we place fires, which would spread fires indiscriminately.

The other problem is that this approach creates illegal firefighter placements, since the branching step does not associate firefighters with fires. To overcome this, before adding a node v to our set F of nodes to be occupied by firefighters, we check the size of a maximum matching between the nodes in $F \cup \{v\}$ and the nodes in B. If there is a matching that includes every vertex in $F \cup v$, then every firefighter can be matched to a unique fire, so putting a firefighter on v does not create an illegal placement. If we delete the edges between two burning nodes or two firefighters, the subgraph induced by $B \cup F \cup v$ is bipartite with at most $2k$ vertices and k^2 edges. A maximum matching in a bipartite graph can be computed in $O(\sqrt{n}m)$ time [6], or in this case in $O(k^{2.5})$ time.

```
**Algorithm** politiciansFirefighting($V, E, B, F, T, k$)
  **if** $k < 0$ **then return** false
  **if** $T$ is empty **then**
    $T \leftarrow \{v \in V \setminus (F \cup B) : v \text{ is adjacent to a node in } B\}$
    **if** $T$ is still empty **then return** true
    **if** $|T| -$ max_match($B, T, E$) $> k$ **then**
      **return** false (more than $k$ fires will spawn this round)
  Choose any $v \in T$
  **if** max_match($B, F \cup \{v\}, E$) $= |F| + 1$ and
    politiciansFirefighting($V, E, B, F \cup \{v\}, T \setminus \{v\}, k$) **then return** true
  **return** politiciansFirefighting($V, E, B \cup \{v\}, F, T \setminus \{v\}, k - 1$)
```

Algorithm politiciansFirefighting runs in time $O(m + k^{2.5}4^k)$, which can be verified as follows. The height of the search tree is bounded by $2k$: Overall we cannot let more than k nodes burn; furthermore we cannot place more than k firefighters because every fire gives us one firefighter to place. This results in a search-tree size of at most $2^{2k} = 4^k$ nodes.[4] The time per node is dominated by the cost of procedure max_match, which computes a maximum matching for a given bipartite graph and takes $O(k^{2.5})$ time.

We have implemented our algorithm to measure the average search-tree size in practice. Our experimental results indicate that, although the size of our search tree is 4^k in the worst-case, in practice the running times are much better.

[4] This worst-case search-tree size for our algorithm is indeed tight up to a polynomial factor. If the number of threatened nodes is exactly twice the number of burning neighbours, the number of legal firefighter placements that must be generated is $\geq \frac{2^{2b}}{2b}$, which is greater than x^{2b} for any $x < 2$ and sufficiently large b.

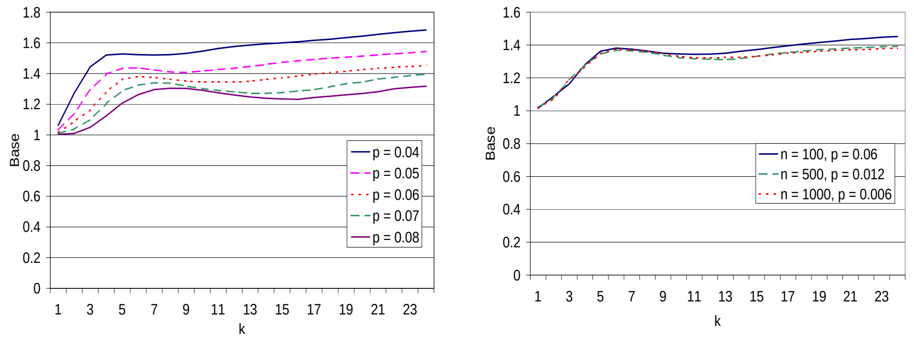

Fig. 3. Left: search-tree size for $n = 100$, $k \leq 25$. Base refers to the x in our x^k search tree size. Right: search-tree sizes for $n = 100, 500, 1000$, $k \leq 25$

For five different densities, we tested 1000 random (connected) graphs from $G_{n,p}$ where n is the number of nodes and p is the probability of any given edge occuring. As shown in Figure 3 (left), running time decreases as graph density increases. This is likely due to nodes in the graph burning more quickly, causing the algorithm to reach a no-answer sooner. Therefore, we concentrated on sparse graphs. We also checked several larger test cases, but as Figure 3 (right) shows, the number of search nodes actually decreases slightly as n increases and relative density is maintained.

References

1. S. Finbow, A. King, G. MacGillivray, and R. Rizzi. The firefighter problem for graphs of maximum degree three. In *Proc. European Conf. on Comb., Graph Theory and Appl.*, 2003.
2. M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. In *Proc. 6th ACM Symp. on the Theory of Comp.*, pp. 47–63, 1974.
3. S. G. Hartke. *Graph-Theoretic Models of Spread and Competition.* PhD thesis, Rutgers University, 2004.
4. B. Hartnell. Firefighter! an application of domination. Presentation at the *24th Manitoba Conf. on Comb. Math. and Comp.*, 1995.
5. B. Hartnell and Q. Li. Firefighting on trees: How bad is the greedy algorithm? *Congressus Numerantium*, 145:187–192, 2000.
6. J. E. Hopcroft and R. M. Karp. A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.
7. F. Roberts. “Challenges for discrete mathematics and theoretical computer science in the defense against bioterrorism” *in Mathematical Modeling Approaches in Homeland Security,* pages 1–34. SIAM Frontiers in Applied Mathematics, 2003.
8. P. Wang and S. Moeller. Fire control in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 41:19–34, 2002.

View publication stats