Online Firefighting on Cactus Graphs

Max Hugen
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Bob Krekelberg
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Alison Hsiang-Hsuan Liu
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

The *firefighting game* is a fundamental problem in theoretical computer science, modeling the containment of a spreading process under limited defensive resources. In each round, an algorithm may protect a set of vertices by placing firefighters on them, preventing the fire from spreading to those vertices. Vertices that are never reached by the fire are said to be *saved*. The goal is to maximize the number of saved vertices. While the offline version has been extensively studied, much less is known about the *competitive complexity* of the online variant, where the underlying graph is known in advance but the number of available firefighters in each round is revealed online.

We study how *graph structure* governs the power of the adversary in the online firefighting game. On trees, the problem is known to be 2-competitive (Coupechoux et al., 2019), a result that relies on a strong structural alignment between the online algorithm and the offline optimal solution throughout the process. We show that this alignment breaks down as soon as cycles are present. In particular, the algorithm and the offline optimal solution may break a cycle differently, or even break different cycles, and therefore operate on fundamentally different residual graphs. As a result, the adversary gains additional leverage by steering the process along residual graph structures that no longer admit a direct comparison between the algorithm and the offline optimal solution.

Our main result shows that the presence of a single cycle already increases the competitive complexity of the firefighting game dramatically. We first show that even on a tadpole graph (a cycle with a tail), no deterministic online algorithm can achieve a competitive ratio better than $\Omega(\sqrt{n})$, where n is the number of vertices. We complement this lower bound with matching upper bounds by designing an $O(\sqrt{n})$-competitive online algorithm for 1-almost trees, that is, graphs obtained from a tree by adding at most one edge. Furthermore, we extend our framework to cactus graphs and prove that, despite the presence of multiple cycles, the competitive complexity remains $\Theta(\sqrt{n})$ as long as the cycles do not share edges. Together, these results yield a tight characterization of the adversarial power induced by cycles with non-overlapping edges. Finally, considering that cactus graphs have treewidth of 2, we study a variant in which firefighters are released in pairs, that is, an even number of firefighters becomes available in each round. Surprisingly, the competitive complexity is significantly reduced to 3-competitive for this setting.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Firefighting game, Online algorithms, Cactus graphs, 1-almost trees

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.70

Related Version *Full Version*: `https://arxiv.org/abs/2509.22277`

1 Introduction

The *online firefighting game* proceeds in rounds as follows. Initially, the algorithm is given a graph with a designated fire source. At the beginning of each round, the algorithm is informed of the number of firefighters available in that round and assigns them to protect unburned vertices. Subsequently, every unprotected neighbor of a burning vertex catches fire, concluding the round. The online algorithm aims to maximize the number of saved vertices, without knowing in advance how many firefighters will be available in future rounds.

We evaluate the performance of an online algorithm for the firefighting game using *competitive analysis*. An algorithm ALG is c-competitive if $\sup_{\mathcal{I}} \frac{\mathrm{OPT}(\mathcal{I})}{\mathrm{ALG}(\mathcal{I})} \leq c$,[1] where $\mathrm{ALG}(\mathcal{I})$ is the number of vertices saved by ALG on instance $\mathcal{I}$, and $\mathrm{OPT}(\mathcal{I})$ is the maximum number of vertices that can be saved with full knowledge of the firefighter sequence. From the complexity perspective, the *competitive ratio of an online problem* is defined as the infimum competitive ratio achievable by any online algorithm for the problem.

Coupechoux et al. [3] showed that the online firefighting game is constant-competitive on trees. Cycles fundamentally change this picture. When a cycle is already threatened by the fire, saving it may require two firefighters, so the algorithm can no longer rely on the greedy choice in every round. Instead, it may have to choose between securing a small immediate gain and breaking a cycle for a potentially much larger but delayed gain. This destroys the tree-based guarantee that a greedy step captures a constant fraction of the optimum. Moreover, breaking a cycle changes the residual graph, and different choices may lead the algorithm and the offline optimal solution to evolve on different residual graphs. By controlling the firefighter sequence, the adversary can exploit this misalignment and steer the algorithm toward an unfavorable residual graph.

Related work

Introduced by Hartnell in 1995 [6], the firefighter problem models the spread of a threat through a network. Applications include limiting misinformation in social or computer networks, immunizing populations, and controlling invasive species. The firefighting game is computationally hard, remaining NP-hard even on restricted graph classes such as bipartite graphs [8] and trees of maximum degree 3 [5].

On the positive side, several approximation algorithms are known for the firefighting game. For example, the greedy algorithm is shown to be a 2-approximation on trees [7]. There also exists a polynomial time approximation scheme for the firefighting game on trees [1].

The online variant of the firefighting game was introduced by Coupechoux et al. [3]. They showed that the greedy algorithm on trees in the online version of the problem is 2-competitive and optimal. Also, in the fractional version of the problem, in which a fraction of a firefighter can be assigned to a vertex to partially protect it, Coupechoux et al. [3] show that the greedy algorithm is 2-competitive on trees. Furthermore, in the special case where the number of firefighters is bounded by 2, Coupechoux et al. [3] propose a ϕ-competitive algorithm on trees, where ϕ is the golden ratio. Later, Demange et al. [4] investigated sufficient conditions for online containment of fires on infinite grid graphs. They show that for an algorithm to contain the fire, there needs to exist a turn t such that the sum of firefighters released until (including) turn t is at least $16t$.

Our contribution

In this work, we study the competitive complexity of the online firefighting game. The 2-competitiveness of the game on trees indicates that the problem is relatively simple in this setting. This raises the question of how the competitive complexity evolves on broader classes of graphs. Surprisingly, we find that adding a single extra edge to a tree already makes the problem no longer constant-competitive.

[1] Note that in this version of the definition of competitive ratio we use for maximization problems, the competitive ratio is always larger than or equal to 1 [2].

Our results show that cycles fundamentally change the online firefighting game. We first prove that even a single cycle already destroys constant competitiveness: on tadpole graphs, every deterministic online algorithm has a competitive ratio of at least $\Omega(\sqrt{n})$ (Theorem 1). We complement this lower bound with a matching upper bound for 1-almost trees (Theorem 7). Then, we extend the result to cactus graphs, showing that additional edge-disjoint cycles do not further increase the competitive complexity. That is, the problem remains $\Theta(\sqrt{n})$-competitive in this class (Theorem 15). Finally, given that cactus graphs have treewidth at most two, we identify a striking contrast in the variant where firefighters arrive in pairs, for which the competitive ratio on cactus graphs drops to a constant (Theorem 16).

Technical overview

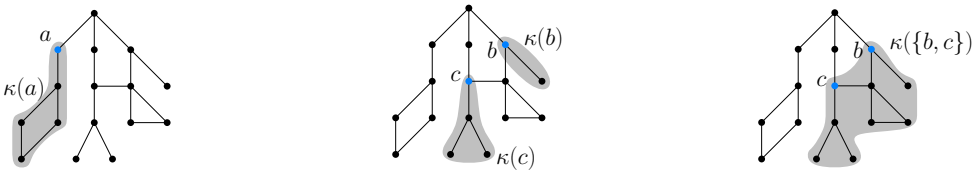
The main obstacle in moving from trees to cactus graphs is that the structural alignment between the online algorithm and the optimal offline solution breaks down once cycles are present. On trees, both solutions can be compared on essentially the same residual graph. In contrast, on graphs with cycles, different choices of cycle vertices may lead to different residual graphs by altering shortest paths. On cactus graphs, this difficulty is even greater, since the two solutions may also choose different root cycles.

Our algorithms balance immediate gain against delayed gain. When a large component can be saved immediately, the algorithm acts greedily. Otherwise, it breaks a carefully chosen cycle to delay the fire, using a tolerance measure to capture how long a breaking point keeps many vertices temporally safe. For cactus graphs, where several root cycles may coexist, we add a cool-down timer whose length reflects the depth of the preserved unburned vertices. This guarantees that if a firefighter arrives before the timer expires, a greedy step saves at least as many vertices as remain on the path defining the timer.

The misalignment between the algorithm's and the offline optimal solution's residual graphs also makes the analysis challenging. To overcome this, we use a time-indexed comparison with a suitably structured offline optimal solution. The key is to partition the vertices saved by the optimum into parts that can be handled independently, in a way that respects the underlying graph structure, and to charge only the exclusive contribution of each part to a carefully chosen subset of vertices protected by the online algorithm. This yields an analysis that does not rely on a direct vertex-by-vertex correspondence between the two solutions, while ensuring that each algorithmic vertex is charged only a constant number of times.

Paper organization

The remainder of the paper is organized as follows. Section 2 introduces the necessary preliminaries. In Section 3, we study 1-almost trees, presenting both the lower bound from a single cycle and an optimal online algorithm together with our analysis framework. Section 4 extends this approach to cactus graphs. In Section 5, we consider the variant in which firefighters arrive in pairs and show that the problem becomes constant-competitive on cactus graphs. Due to space constraints, we provide only proof sketches here; full proofs are available in the full version.



(a) The covered set of a protected non-cycle vertex.

(b) The covered sets of protected cycle vertices.

(c) The covered set of a set of protected (cycle) vertices.

Figure 1 Examples of covered sets.

2 Preliminaries

Graph and graph classes

Let $G = (V, E)$ be an undirected graph. A *(simple) cycle* is a sequence of distinct vertices $(u_1, u_2, \cdots, u_k) \subseteq V$ with $k \geq 3$ such that $(u_i, u_{i+1}) \in E$ for all $1 \leq i \leq k-1$, and $(u_1, u_k) \in E$ are edges. [2] Given any graph G, we denote $V(G)$ as the set of vertices in G if it is not specified. The graph G is a 1-*almost-tree* if it is isomorphic to a connected graph with $|V(G)|$ vertices and $|V(G)|$ edges. Equivalently, a 1-almost-tree can be seen as a tree with one extra edge and has exactly one cycle.[3] A graph $G = (V, E)$ is a *cactus* graph if every edge $e \in E$ is contained in at most one cycle. In other words, in cactus graphs, two cycles can share at most one vertex. In this paper, we say that a vertex is a *cycle vertex* if it is in at least one cycle, and a *non-cycle* vertex otherwise. Moreover, a cycle is a *root cycle* if it contains the fire source r.

Online firefighting game

In an *online firefighting game*, the instance is $\mathcal{I} = (G, r, (f_i)_{i \geq 1})$, where $G = (V, E)$ is a graph with the *fire source* or *root* $r \in V$, and $f_i \geq 0$ is the number of available firefighters in round i. We denote n as the number of vertices in G (that is, $n = |V(G)|$ using our notation). Let the *(open) neighborhood* of a vertex $v \in V$ be the set of all vertices adjacent to v and denoted by $N(v)$. That is, $N(v) = \{u \mid (u, v) \in E\}$. In each round i of the firefighting game, the online algorithm learns the value of f_i and *protects* unburned vertices by assigning the firefighters to them. Then, the fire spreads to unprotected vertices in $N(v)$ for all burned vertices v, and it ends this round. In this work, we consider the cases where G is a 1-almost-tree (Section 3) or a cactus graph (Section 4).

Status of a vertex

At any time in the firefighting game, a vertex can be *protected*, *burned*, *saved*, or *available*. A vertex is *protected* indefinitely if there is a firefighter assigned to it. A vertex is *saved* if there is at least one protected vertex on every path from r to v. A vertex v such that there exists a path from r to v containing no protected vertices is *burned* if the fire has reached v and *available* otherwise. Using these terminologies, the algorithm's goal is equivalent to saving as many vertices as possible.

---

[2] For accommodating other notations used in the paper, we use (u, v) as edges. Since the graph discussed in this work is undirected, $(u, v) = (v, u)$.

[3] By a slight abuse of terminology, we define a 1-almost tree as a graph containing at most one cycle.

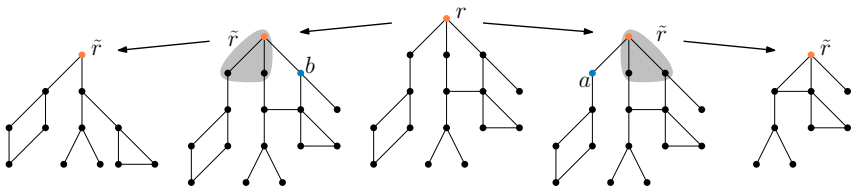


Figure 2 The middle figure shows the initial graph, with r as the fire source. Adjacent figures show the fire spreading, with a or b protected. The outer figures show the reduced graphs.

Covered set and gain of an algorithm

A vertex u is *covered* by v if every path from u to r contains vertex v. Conversely, all the vertices covered by v form a *covered set* of v, denoted by $\kappa(v)$. More generally, a vertex u is covered by a set of vertices S if every path from u to r contains at least one vertex in S, and the covered set of S, $\kappa(S)$, is defined analogously.[4] Intuitively, the covered set of a set of vertices S can be seen as the set of vertices that are saved by protecting all vertices in S (before fire spreads to them). Equivalently, it is the set of vertices that are saved by placing firefighters on each vertex in S before the fire starts spreading. Note that by this definition, $\kappa(S)$ can be larger than $\bigcup_{v\in S}\kappa(v)$, as illustrated in Figure 1. The *weight* of a set of vertices S, $w(S) = |\kappa(S)|$, is the size of its covered set.

Let Ψ^{A} denote the set of all vertices protected by our algorithm ALG. By definition, $\kappa(\Psi^{\mathrm{A}})$ is the set of vertices saved by ALG, and $w(\Psi^{\mathrm{A}})$ is the final gain of ALG (i.e., the total number of vertices saved by ALG). Similarly, Ψ^* denotes the set of all vertices protected by the offline optimal solution OPT, and $\kappa(\Psi^*)$ and $w(\Psi^*)$ are defined symmetrically.

Graph reduction

We follow the reduction procedure on online firefighting on trees by Coupechoux et al. [3]. At the beginning of a round, the algorithm sees the graph G with a fire source r. At the end of the round, after the algorithm's decisions of protecting a set of available vertices P, and the fire spreading to all unprotected neighbors of r, all burned vertices are contracted with r into a new fire source r'. Moreover, all vertices in $\kappa(P)$ are removed from the graph. The new graph G' with the new fire source r' is the input to the algorithm in the next round. By this reduction procedure, the algorithm always sees exactly one burned vertex, which is the fire source, and all other vertices are available. An example of the reduction procedure is illustrated in Figure 2.

Note that the resulting residual graph is sensitive to the protected vertex. If the algorithm protects a vertex different from the offline optimal algorithm, the graph produced by the reduction procedure for each algorithm might differ substantially. For an illustration of this, compare the difference after protecting vertex a and vertex b in Figure 2.

For analysis, we first observe that there exists a *non-redundant* offline optimal solution, where every cycle contains at most two protected vertices. From now on, we compare our algorithms' performance against a non-redundant offline optimal solution.

[4] Naturally, we only consider $S \subseteq V \setminus \{r\}$.

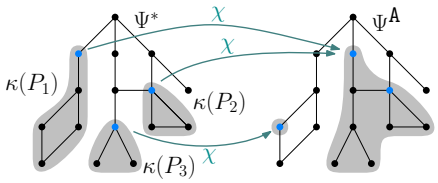


Figure 3 Under some charging function χ, $P_1, P_2 \subset \Psi^*$ map to the same subset of Ψ^{A}: the vertex protected by ALG when OPT protects a vertex in P_i ($i \in 1, 2$), plus one additional vertex on the same cycle. The set P_3 maps to the vertex protected by ALG at the corresponding time.

Charging scheme for analysis

In the analysis, we first show that for the case of cactus graphs, there always exists a partition of the set Ψ^* of vertices protected by OPT into $P_1, P_2, \cdots$ such that $\kappa(P_i) \cap \kappa(P_j) = \emptyset$ if $i \neq j$ (Lemma 3). Then, we focus on the *exclusive covered set* of each P_i, which is defined as $\kappa'(P_i) = \kappa(P_i) \setminus \kappa(\Psi^{\mathrm{A}})$, and the *exclusive weight* $w'(P_i) = |\kappa'(P_i)|$. For each P_i, the goal is to find some subset of Ψ^{A} and charge the gain of protecting P_i to the gain of protecting this subset. More formally, define a charging function $\chi : \{P_i\}_i \to 2^{\Psi^{\mathrm{A}}}$ that maps each P_i to a subset of vertices protected by the algorithm (see Figure 3 for an example). The main challenge is to find a proper partition and a proper charging function such that for every P_i, $w'(P_i) \leq \sqrt{n} \cdot \alpha_i \cdot w(\chi(P_i))$ for some constant α_i (i.e., the inequality labeled by $(\star)$ in Inequality (1)). Then, let $\alpha_{\max}$ be the largest α_i and β be the maximum number of times an element $\psi^{\mathtt{A}} \in \Psi^{\mathrm{A}}$ is charged by parts P_i:

$$\begin{aligned} \mathrm{OPT}(\mathcal{I}) &\overset{\text{Lemma 3}}{=} \sum_i w(P_i) \leq w(\Psi^{\mathrm{A}}) + \sum_i w'(P_i) \overset{(\star)}{\leq} w(\Psi^{\mathrm{A}}) + \sqrt{n} \cdot \sum_i \alpha_i \cdot w(\chi(P_i)) \\ &\overset{\text{amortization}}{\leq} w(\Psi^{\mathrm{A}}) + \sqrt{n} \cdot \alpha_{\max} \cdot \beta \cdot w(\Psi^{\mathrm{A}}) = (1 + \sqrt{n} \cdot \alpha_{\max} \cdot \beta) \cdot \mathrm{ALG}(\mathcal{I}) \end{aligned} \quad (1)$$

3 Warm-up: 1-almost-tree

In this section, we show that the online firefighting problem is significantly harder with the presence of a cycle. Specifically, we will show that no deterministic online algorithm can achieve $o(\sqrt{n})$-competitiveness on tadpole graphs, the class of graphs obtained by joining a cycle graph to a path graph. We then propose an optimal algorithm for 1-almost trees.

▶ **Theorem 1.** *No deterministic online algorithm can achieve $o(\sqrt{n})$-competitiveness for the online firefighting game on tadpole graphs.*

Proof sketch. Consider an arbitrary deterministic online algorithm ALG. Let G be a tadpole graph, consisting of a cycle of α vertices and a path of β vertices, joined at the fire source.

If ALG protects a path vertex, the adversary releases one more firefighter, and OPT protects the cycle. Otherwise, no firefighters are released, and OPT protects the path.

Thus the competitive ratio is at least $\min\{\beta, \frac{\alpha}{\beta}\}$. Balancing the two terms gives $\alpha = \Theta(\beta^2)$. Since $n = \Theta(\alpha + \beta) = \Theta(\beta^2)$, the ratio is $\Omega(\sqrt{n})$. ◀

Theorem 1 shows that a wise online algorithm might consider breaking a cycle when the heaviest component is a cycle, and there are not enough firefighters to protect it completely. Next, we provide mathematical support for efficiently breaking a cycle.

3.1 Breaking cycles efficiently

Temporal safeness and tolerance

Let $\texttt{dist}(G, u, v)$ denote the shortest distance between vertices u and v in graph G. We denote $count(G, d)$ as the number of vertices in G that are *temporally safe* with respect to d. Formally, $count(G, d) = |\{v \in V \mid \texttt{dist}(G, r, v) \geq d\}|$, which is the number of vertices in G that have the shortest distance to r of at least d. Note that $count(G, d_2) \leq count(G, d_1)$ for any $0 \leq d_1 \leq d_2$ for all cactus graphs G.

Given a root cycle C and a vertex u on it, we denote $T(C \backslash u)$ as the subgraph induced by $\{r\} \cup \kappa(C)$ with u being protected and $\kappa(u)$ being removed. Formally, $T(C \backslash u) = G[\{r\} \cup \kappa(C) \setminus \kappa(u)]$. Then, we define the *tolerance* of a cycle vertex u in a root cycle C with respect to a natural number m, $\texttt{tol}(u, C, m)$, as the furthest the fire can spread in the induced subgraph $T(C \backslash u)$ while still leaving at least m vertices temporally safe. Formally, $\texttt{tol}(u, C, m) = \max\{d \in \mathbb{N} \mid count(T(C \backslash u), d) \geq m\}$.

Cycle-break as a delay mechanism

Protecting a single vertex on a cycle "breaks" the cycle, transforming it into tree branches in the reduced graph. See Figure 2 for an example where the protected vertex b breaks the 5-cycle. Our algorithms use the tolerance of vertices to decide on which vertex to break a cycle. Intuitively, if the algorithm breaks the cycle C in a way that maximizes the number of rounds that at least $m = \sqrt{w(C)}$ vertices remain unburned, even in the case where the offline optimal solution manages to save the entire cycle C, the algorithm has the hope to secure at least $\sqrt{w(C)}$ vertices itself, and thus remain $O(\sqrt{n})$-competitive.

3.2 ALG_{A}: The algorithm for 1-almost-trees

By the reduction procedure, we focus on a single round i of the firefighting game with a graph $\widetilde{G}$ with a fire source $\tilde{r}$ and f_i available firefighters. In this round i, the algorithm ALG_{A} iteratively assigns firefighters to vertices in $N(\tilde{r}) \cup \widetilde{C}$ until there are no firefighters left.

In each iteration, if $\tilde{r}$ is not involved in any cycle (that is, there is no root cycle), ALG_{A} protects the heaviest neighbor of $\tilde{r}$. When $\widetilde{G}$ has a root cycle, ALG_{A} makes decisions between protecting the heaviest neighbor of $\tilde{r}$, protecting the root cycle with two firefighters, or breaking the root cycle. At the end of one iteration, the protected vertices v and $\kappa(v)$ are removed from $\widetilde{G}$, and the number of available firefighters is reduced accordingly.

The fire source $\tilde{r}$ is not in any cycle

The algorithm ALG_{A} makes its decisions greedily if there is no root cycle in the input graph $\widetilde{G}$. That is, it assigns the firefighters to the first f_i "heaviest" neighbors of $\tilde{r}$ that have the highest weights. Formally, the firefighters are assigned iteratively to unprotected $v \in N(\tilde{r})$ with the highest $w(v)$.

The fire source $\tilde{r}$ is in a root cycle $\widetilde{C}$, and there are at least 2 firefighters

In the case where the input graph $\widetilde{G}$ has a root cycle $\widetilde{C}$, we consider the unprotected vertices in $N(\tilde{r}) \cup \widetilde{C} = \{v_1, v_2, \cdots\}$ where $w(v_1) \geq w(v_2) \geq \cdots$. When there are at least two firefighters available, we protect the cycle using two firefighters if $w(\widetilde{C}) > w(v_1) + w(v_2)$. Otherwise, we protect v_1 using one firefighter, and the procedure iteratively continues (with v_1 removed from the candidates of the vertices that need to be protected).

The fire source $\tilde{r}$ is in a root cycle $\widetilde{C}$, and there is only one firefighter

When there is a root cycle $\widetilde{C}$ in the input graph $\widetilde{G}$ and there is only one firefighter, the algorithm protects v_1 if $w(v_1) \geq \sqrt{w(\widetilde{C})}$. Otherwise, when the immediate gain from protecting v_1 is not significant enough, the algorithm executes the `Break-cycle` procedure by protecting the vertex $u \in N(\tilde{r}) \cap \widetilde{C}$ with the largest $\texttt{tol}(u, \widetilde{C}, \sqrt{w(\widetilde{C})})$ (tolerance with respect to $\sqrt{w(\widetilde{C})}$ vertices)[5].

3.3 Framework of competitiveness analysis

Quality of breaking cycle at a neighbor of $\tilde{r}$

Consider the scenario where there is a root cycle $\widetilde{C}$, and there is only one firefighter available. We show that by setting the desired number of unburned vertices $m = \sqrt{w(\widetilde{C})}$, breaking $\widetilde{C}$ by protecting the vertex in $N(\tilde{r}) \cap \widetilde{C}$ with higher tolerance is always "beneficial":

▶ **Lemma 2** (Quality of `Break-cycle`). *Let $\widetilde{G}$ be the input graph with a root cycle $\widetilde{C} = (r, u_1, u_2, \cdots, u_p)$. Let $\hat{u}$ be the vertex protected by the* `Break-cycle` *procedure. For every vertex u_h and every $d \in \mathbb{N}$, $\frac{count(T(\widetilde{C} \setminus u_h), d)}{count(T(\widetilde{C} \setminus \hat{u}), d)+1} \leq 2\sqrt{w(\widetilde{C})}$.*

Proof sketch. Let $d_{\max} = \texttt{tol}(\hat{u}, \widetilde{C}, \sqrt{w(\widetilde{C})})$. When $d \leq d_{\max}$, the lemma is proven by the monotonicity of the *count* function and the fact that the whole cycle has weight $w(\widetilde{C})$. On the other hand, when $d > d_{\max}$, we first consider the number of vertices that are temporally safe with respect to d after breaking the cycle at u_h. We can show that $count(T(\widetilde{C} \setminus u_h), d) \leq \max_{i \in \{1,p\}} \{count(T(\widetilde{C} \setminus u_i), d) + w(u_i)\}$. Then, noticing that `Break-cycle` only happens when $w(v) \leq \sqrt{w(\widetilde{C})}$ for any vertex v, $count(T(\widetilde{C} \setminus u_h), d) \leq \max_{i \in \{1,p\}} \{count(T(\widetilde{C} \setminus u_i), d) + w(u_i)\} < \sqrt{w(\widetilde{C})} + \sqrt{w(\widetilde{C})} = 2\sqrt{w(\widetilde{C})}$, where the last inequality is from the fact that the algorithm chooses $\hat{u} = \arg\max_{i \in \{1,p\}} \texttt{tol}(u_i, \widetilde{C}, \sqrt{w(\widetilde{C})})$. ◀

Cycle-respecting partition

We want to partition Ψ^* into disjoint parts such that the covered sets of different parts do not intersect. For later generalization to the cactus graphs case, we provide a guideline for partitioning: *cycle-respecting partition*. Given the input graph G in which any two cycles share at most one vertex, a partition $\mathcal{P}^* = \{P_1, P_2, \cdots\}$ is *cycle-respecting* if for every cycle C in G, the vertices in $C \cap \Psi^*$ belong to the same part $P_i \subseteq \mathcal{P}^*$. We show that if $\mathcal{P}^*$ is cycle-respecting, the covered sets of the parts are disjoint, and thus the competitive ratio of each part P_i can be analyzed separately.

▶ **Lemma 3.** *Let G be a cactus graph with fire source r, and Ψ^* is the subset of vertices protected by OPT. Suppose $\mathcal{P}^* = \{P_1, P_2, \cdots, P_m\}$ is a partition of Ψ^* that is cycle-respecting. Then, $\kappa(\Psi^*) = \bigcup_{i=1}^{m} \kappa(P_i)$.*

[5] Intuitively, this `Break-cycle` procedure protects the cycle vertex that maximizes how far the fire can spread while still leaving at least $\sqrt{w(\widetilde{C})}$ vertices unburned

Time indexing

Let $\Psi^{\mathtt{A}} = (\psi_1^{\mathtt{A}}, \psi_2^{\mathtt{A}}, \cdots)$ be the set of vertices protected by our algorithm, where the vertex $\psi_t^{\mathtt{A}}$ is the t-th vertex protected by the algorithm according to the algorithm description. We say that t is the *time label* of the vertex v if $v = \psi_t^{\mathtt{A}}$. In short, we say that $\psi_t^{\mathtt{A}}$ is protected *at time* t. Note that two vertices can be protected by our algorithms "simultaneously" to save a whole cycle. In this case, these two vertices naturally have two consecutive time labels.

We extend the time labeling to the set Ψ^* of vertices protected by OPT. Formally, let $\psi_{t+1}^{\mathtt{A}}, \psi_{t+2}^{\mathtt{A}}, \cdots, \psi_{t+f_i}^{\mathtt{A}}$ be the vertices protected by the algorithm in round i. We fix an arbitrary one-to-one mapping between $\psi_{t+1}^{\mathtt{A}}, \cdots, \psi_{t+f_i}^{\mathtt{A}}$ and the f_i vertices protected by the offline optimal solution in the same round (note that the algorithm cannot finish the game later than the offline optimal solution). We denote by ψ_t^* the vertex protected by the offline optimal solution, if it is the image of $\psi_t^{\mathtt{A}}$ under the one-to-one mapping. Thus, we also order the vertices protected by the offline optimal solution $\psi_1^*, \psi_2^*, \cdots$. We say that $\psi_t^{\mathtt{A}}$ and the corresponding ψ_t^* are protected *at the same time*.

We further extend this time labeling to the input graph $G_t^{\mathtt{A}}$, which is the graph from the algorithm's perspective right before the vertex $\psi_t^{\mathtt{A}}$ is protected. Note that if $\psi_t^{\mathtt{A}}$ and $\psi_{t+1}^{\mathtt{A}}$ are protected in the same round, $G_{t+1}^{\mathtt{A}}$ is a subgraph of $G_t^{\mathtt{A}}$ with $\kappa(\psi_t^{\mathtt{A}})$ removed (see the description of the algorithms for how these graphs change throughout the game). Moreover, consider the case where both $G_t^{\mathtt{A}}$ and $G_{t+1}^{\mathtt{A}}$ have a root cycle corresponding to the same cycle in the original graph, and $\psi_t^{\mathtt{A}}$ and $\psi_{t+1}^{\mathtt{A}}$ are not protected in the same round. The root cycle in $G_{t+1}^{\mathtt{A}}$ is smaller than the one in $G_t^{\mathtt{A}}$, because of the graph reduction procedure. We also denote $\kappa_t(S)$ as the set of vertices covered by the set of vertices $S \subseteq V(G_t^{\mathtt{A}})$ and $w_t(S) = |\kappa_t(S)|$ as the number of vertices saved correspondingly in graph $G_t^{\mathtt{A}}$.

3.4 Competitive analysis of $\mathrm{ALG}_{\mathtt{A}}$

Partition $\mathcal{P}^*$ of Ψ^*

We partition Ψ^* into the three parts:

- P^a: **Cycle vertices.** All vertices in Ψ^* that are cycle vertices in the original graph G,
- P^b: **Non-cycle vertices and available to $\mathrm{ALG}_{\mathtt{A}}$.** All vertices $\psi_t^* \in \Psi^*$ that are available (unburned) in $G_t^{\mathtt{A}}$, and
- P^c: **Non-cycle vertices and not available to $\mathrm{ALG}_{\mathtt{A}}$.** All vertices $\psi_t^* \in \Psi^*$ that are burned or saved in $G_t^{\mathtt{A}}$.

Charging function

For the analysis, we define the charging function $\chi : \mathcal{P}^* \to 2^{\Psi^{\mathrm{A}}}$ that charges each part of $\mathcal{P}^*$ to a subset of vertices protected by $\mathrm{ALG}_{\mathtt{A}}$. In the case of $\mathrm{ALG}_{\mathtt{A}}$, we simply charge every part $P^{\cdot} \in \mathcal{P}^*$ to the entire set Ψ^{A} by defining $\chi(P^{\cdot}) = \Psi^{\mathrm{A}}$. That is, we charge each part to Ψ^{A} individually. Note that since we charge each part to Ψ^{A} directly, the analysis framework is a bit easier than the general one.

In the following, we focus on each part $P^{\cdot} \in \mathcal{P}^*$ and show that the size of the exclusive covered set of $P^{\cdot}$ is at most $O(\sqrt{n})$ times the size of the covered set of Ψ^{A}, where n is the total number of vertices in the original graph G.

▶ **Lemma 4** (Cycle vertices in Ψ^*). *Given the part P^a in $\mathcal{P}^*$, $w'(P^a) \leq \sqrt{n} \cdot w(\Psi^A)$, where n is the number of vertices in the input graph G.*

Proof sketch. Let t be the time (label) when OPT protects the first cycle vertex ψ_t^* in P^a. We consider the graph G_t (the graph right before the corresponding vertex $\psi_t^{\mathtt{A}}$ was protected by $\mathrm{ALG}_{\mathtt{A}}$ from the algorithm's perspective), where the heaviest vertex in $N(r) \cap \widetilde{C}$ is $v_1^{(t)}$ (note that $v_1^{(t)}$ can also be a cycle vertex). If $\mathrm{ALG}_{\mathtt{A}}$ has broken the cycle before time t, since the protection on u_c has broken the cycle C into two arcs A_1 and A_2 by time t and $\mathrm{ALG}_{\mathtt{A}}$ saves $v_1^{(t)}$ when there is no cycle, then we can argue that $w(\Psi^{\mathrm{A}}) \geq w'(P^a)/2$.

In contrast, if $G_t^{\mathtt{A}}$ still has a root cycle $\widetilde{C}$, $w'(P^a) \leq w_t(\widetilde{C})$. We consider the different scenarios on how $\mathrm{ALG}_{\mathtt{A}}$ selects $\psi_t^{\mathtt{A}}$ in $G_t^{\mathtt{A}}$. Under the definition of $\mathrm{ALG}_{\mathtt{A}}$, $w(\Psi^{\mathrm{A}}) \geq \sqrt{w_t(\widetilde{C})}$ if $\psi_t^{\mathtt{A}}$ is protected as the heaviest vertex or protecting the heaviest cycle with another vertex. If $\psi_t^{\mathtt{A}}$ is chosen to break the root cycle $\widetilde{C}$, $w_t(v)^2 < w_t(\widetilde{C})$ for all $v \in G_t^{\mathtt{A}}$. Then, even when OPT saves $\widetilde{C}$ while $\mathrm{ALG}_{\mathtt{A}}$ only protect one vertex on $\widetilde{C}$, $w'(P^a) \leq \sqrt{w_t(\widetilde{C})} \cdot w(\Psi^{\mathrm{A}})$ since the other firefighter OPT uses to protect a cycle vertex on $\widetilde{C}$ is also available to $\mathrm{ALG}_{\mathtt{A}}$, and $\mathrm{ALG}_{\mathtt{A}}$ always make the greedy choice when there is no cycle in the graph. ◀

▶ **Lemma 5** (Non-cycle vertices in Ψ^* that are available to $\mathrm{ALG}_{\mathtt{A}}$)**.** *Consider the part P^b in $\mathcal{P}^*$, which consists of all non-cycle vertices $\psi_t^* \in \Psi^*$ that are available in $G_t^{\mathtt{A}}$. Then $w'(P^b) \leq \sqrt{n} \cdot w(\Psi^A)$, where n is the number of vertices in the input graph G.*

Proof sketch. Similar to the proof of Lemma 4, we analyze the behavior of $\mathrm{ALG}_{\mathtt{A}}$ on deciding which vertex to protect by $\psi_t^{\mathtt{A}}$ corresponding to ψ_t^* in $P^{(b)}$. We show that if $\psi_t^{\mathtt{A}}$ is protecting the heaviest vertex or protecting the heaviest cycle with another vertex $\psi_{t+1}^{\mathtt{A}}$, $w(\psi_t^*) \leq w(\psi_t^{\mathtt{A}})$ or $\psi_{t+1}^{\mathtt{A}}$, $w(\psi_t^*) \leq w(\Psi^{\mathrm{A}})$. If $\psi_t^{\mathtt{A}}$ is breaking the cycle, $w'(\psi_t^*) \leq \sqrt{w_t(\widetilde{C})} \leq \sqrt{w_t(\widetilde{C})} \cdot w(\Psi^{\mathrm{A}})$. ◀

▶ **Lemma 6** (Non-cycle vertices in Ψ^* that are unavailable to $\mathrm{ALG}_{\mathtt{A}}$)**.** *Consider the part P^c in $\mathcal{P}^*$, which consists of all non-cycle vertices $\psi_t^* \in \Psi^*$ that are unavailable in $G_t^{\mathtt{A}}$. Then $w'(P^c) \leq (4\sqrt{n}+1) \cdot w(\Psi^A)$, where n is the number of vertices in the input graph G.*

Proof sketch. Consider the vertex $\psi_t^* \in P^c$. By the definition of P^c, ψ_t^* is unavailable in $G_t^{\mathtt{A}}$. Since the fire spread at constant speed in the graph, ψ_t^* is unavailable to $\mathrm{ALG}_{\mathtt{A}}$ because there is a cycle C intersecting every path from ψ_t^* to r in the original graph G, and OPT protected a cycle vertex on C and the shortest path from ψ_t^* to r at time $s < t$, while $\mathrm{ALG}_{\mathtt{A}}$ did not break the cycle or broke the cycle in a way different with OPT. Then, we consider the graph $G_s^{\mathtt{A}}$, which has a root cycle $\widetilde{C}$ from the algorithm's perspective, and analyze the behavior of $\mathrm{ALG}_{\mathtt{A}}$ at time s. The cases of $\psi_s^{\mathtt{A}}$ protecting the heaviest vertex or the heaviest cycle (together with another vertex) are similar to the proof in Lemma 5. In the third case where $\mathrm{ALG}_{\mathtt{A}}$ also breaks the cycle at time s, the graphs $\mathrm{ALG}_{\mathtt{A}}$ and OPT play on are two trees. We show that after $\mathrm{ALG}_{\mathtt{A}}$ finishes the game, OPT can save at most $count(T(\widetilde{C} \backslash \psi_s^*), \delta)$ additional vertices by protecting vertices in P^c. Meanwhile, since $\mathrm{ALG}_{\mathtt{A}}$ finished the game by this time, $\mathrm{ALG}_{\mathtt{A}}$ has covered at least $count(T(\widetilde{C} \backslash \psi_s^{\mathtt{A}}), \delta) + w_s(\psi_s^{\mathtt{A}})$. The lemma is then proven using Lemma 2. ◀

▶ **Theorem 7.** *There is a $(5\sqrt{n}+2)$-competitive algorithm $ALG_{\mathtt{A}}$ for the online firefighting game on 1-almost-trees, which is asymptotically optimal.*

Proof. By Lemmas 4, 5, and 6, $\sum_{P^{(\cdot)} \in \mathcal{P}^*} w'(P^{(\cdot)}) \leq (5\sqrt{n}+1) \cdot w(\Psi^{\mathrm{A}})$. Therefore, $\frac{\mathrm{OPT}(\mathcal{I})}{\mathrm{ALG}(\mathcal{I})} = \frac{w(\Psi^*)}{w(\Psi^{\mathrm{A}})} \leq \frac{w'(\Psi^*)}{w(\Psi^{\mathrm{A}})} + 1 = \frac{\sum_{P^{(\cdot)} \in \mathcal{P}^*} w'(P^{(\cdot)})}{w(\Psi^{\mathrm{A}})} + 1 \leq \frac{(5\sqrt{n}+1) \cdot w(\Psi^{\mathrm{A}})}{w(\Psi^{\mathrm{A}})} + 1 = 5\sqrt{n} + 2$. ◀

4 Cactus graphs

Recall that in the 1-almost tree case, we use $T(C\backslash u)$ and $\texttt{tol}(u, C, m)$ to measure the efficiency of breaking the root cycle C by protecting u. In the cactus graph case, we extend the notation to $T_e(C\backslash(u, v))$, where (u, v) is an edge of the root cycle C, to denote the subgraph induced by $\{r\} \cup \kappa(C)$ with the edge (u, v) removed. Similarly, we also extend the notion to $\texttt{tol}_\texttt{e}((u, v), C, m) = \max\{d \in \mathbb{N} \mid \mathit{count}(T_e(C\backslash(u, v)), d) \geq m\}$.

4.1 $\text{ALG}_\texttt{C}$: Algorithm for cactus graphs

Similar to the previous case, $\text{ALG}_\texttt{C}$ proceeds in rounds, each consisting of several iterations. In each iteration, if there are enough firefighters to protect the heaviest component adjacent to the fire source, $\text{ALG}_\texttt{C}$ makes the greedy choice. Moreover, if only one firefighter is available but the greedy choice guarantees a gain of at least $\sqrt{w(\widetilde{C}_1)}$, where $\widetilde{C}_1$ is the heaviest root cycle in the current graph, the algorithm still follows the greedy choice.

The critical case occurs when only one firefighter is available, and the greedy choice does not guarantee a sufficiently large immediate gain. In this situation, as in the strategy for 1-almost trees, $\text{ALG}_\texttt{C}$ breaks a root cycle C by protecting a vertex u on the cycle that maximizes $\texttt{tol}_\texttt{e}(u, \widetilde{C}, \sqrt{w(\widetilde{C}_1)})$, thereby ensuring that at least $\sqrt{w(\widetilde{C}_1)}$ vertices lie as far from the fire as possible (in terms of distance from $\tilde{r}$). Unlike in 1-almost trees, however, a cactus graph may contain multiple root cycles. A difficulty arises because if the algorithm repeatedly breaks cycles due to the lack of immediate gain, it may fail to secure many vertices that could otherwise be saved. To address this, we introduce a *cool-down timer* mechanism: once a cycle is broken, the algorithm enters a cool-down period during which it must follow the greedy choice whenever new firefighters become available.

Cool-down mechanism and ImprovedBreak

We first find the cycle vertex adjacent to the fire source $\tilde{r}$ that has the largest tolerance in the sense that if the edge between it and $\tilde{r}$ is removed, the distance that the fire can spread while keeping $\sqrt{w(\widetilde{C}_1)}$ vertices remaining unburned is maximized. Let $\widetilde{C}_*$ be the cycle containing this vertex u_* with the largest tolerance $d_{\max}$. Now the ImprovedBreak algorithm tries to find an alternative cycle vertex on $\widetilde{C}_*$ while keeping the invariant "*at least* $\sqrt{w(\widetilde{C}_1)} - w(u)$ *vertices in* $T(\widetilde{C}_*\backslash(r, u_*))$ *remain unburned for* $d_{\max}$ *rounds*". This alternative chooses a break-cycle vertex farther from the fire source, while exposing only a limited number of additional vertices to the risk of being burned compared to breaking at a neighbor of $\tilde{r}$. The setting of the cool-down timer ensures that whenever a firefighter is released before the timer expires, the greedy choice protects at least as many vertices as the remaining available ones on the path that defines the timer.

Quality of the choice of cycle-breaking vertex $\hat{u}$

We first show that ImprovedBreak indeed returns a cycle vertex $\hat{u}$ with the desired properties. Using this, we show that the number of vertices that remain at distance $d_{\max}$ from the fire source by protecting $\hat{u}$ is within a bounded factor of this measure by breaking any (other) cycle in any other way.

Algorithm 1 Procedure ImprovedBreak.

Procedure ImprovedBreak(G, r, η)
  Output: a cycle vertex $\hat{u}$ and a cool-down distance Cool-down.
  Let $\widetilde{C}_1, \widetilde{C}_2, \cdots$ be root cycles with weight at least η, where $w(\widetilde{C}_1) \geq w(\widetilde{C}_2) \geq \cdots$
  Let $A \leftarrow \bigcup_h \{\, u \in N(r) \cap \widetilde{C}_h : w(\widetilde{C}_h) - w(u) \geq \sqrt{w(\widetilde{C}_1)} \,\}$.
  Let $u_* \leftarrow \arg\max_{u \in A} \mathtt{tol_e}(u, \widetilde{C}_h, \sqrt{w(\widetilde{C}_1)})$ where $\widetilde{C}_h$ is the cycle containing u.
  Let $\widetilde{C}_* = (r, u_1, \ldots, u_p)$ with $u_1 = u_*$. And let $d_{\max} \leftarrow \mathtt{tol_e}(u_*, \widetilde{C}_*, \sqrt{w(\widetilde{C}_1)})$.
  for $\hat{u} = u_1, u_2, \cdots$ **do**
    if *there is* $v \in \kappa(\hat{u})$ *such that* $\mathtt{dist}(T_e(\widetilde{C}_* \backslash (u_*, r)), r, v) \geq d_{\max}$ **then**
      return $\hat{u}$, $\mathtt{dist}(T_e(\widetilde{C}_* \backslash (u_*, r)), r, \hat{u})$

▶ **Lemma 8** (Secured alternative cycle break). *Assume that the algorithm ALG_C protects $\hat{u}$ to break the cycle $\hat{C}$. Then, for any $d \in \mathbb{N}$, any root cycle $\widetilde{C}$, and every vertex $u \in \widetilde{C}$, $\frac{count(T(\widetilde{C} \backslash u), d)}{count(T(\widetilde{C} \backslash \hat{u}), d) + w(\hat{u})} \leq 2\sqrt{n}$, where n is the number of vertices in the original input graph G.*

Next, we show that during the cool-down period set by ImprovedBreak, the number of vertices saved by the forced greedy choice, together with the protection of vertex $\hat{u}$, is at least a $\frac{1}{\sqrt{n}}$ fraction of the number of vertices saved by protecting any possible pair of vertices.

▶ **Lemma 9** (Protection quality in cool-down period). *Let $\hat{u}$ and Cool-down be the values returend by ImprovedBreak in round i. Let i' be the next round with available firefighters. If $i' - i \leq$ Cool-down, let u' be the first vertex protected by ALG_C in round i'. For arbitrary vertices x and x' that are available in round i and round i', respectively, $\frac{w(\{x, x'\})}{w(\{\hat{u}, u'\})} \leq 2\sqrt{n}$.*

4.2 Competitive analysis of $\mathsf{ALG_C}$

For analyzing $\mathrm{ALG_C}$ on cactus graphs, we first introduce the concept of *delay*. A vertex v is *delayed* by another vertex p when p is protected, v is not covered, but the time that fire reaches v is strictly delayed. That is, p is v's *delayer* if p is on every shortest path from r to v in G and $v \notin \kappa(p)$. We generalize this concept and say that given a vertex p, we define its *delayed set* $\mathtt{D}(p)$ by the set of all vertices that are delayed by p. For a set S of vertices, its delayed set $\mathtt{D}(S)$ is defined by $\bigcup_{p \in S} \mathtt{D}(p)$. Furthermore, given $\psi^* \in \Psi^*$, we denote $\mathtt{D}(\psi^*) \cap \Psi^*$ as $\mathtt{D}_{\Psi^*}(\psi^*)$. That is, $\mathtt{D}_{\Psi^*}(\psi^*)$ is the set of vertices protected by OPT and delayed by ψ^*.

Recall that in algorithm $\mathrm{ALG_C}$, at most three vertices can be protected on the same cycle. We call such cycle vertices *peers*.

Partition $\mathcal{P}^*$ of Ψ^*

Given the set Ψ^* of vertices protected by the non-redundant OPT. We partition Ψ^* into three *types* of parts, where there can be multiple parts of Type b or c.

Type a: **Non-cycle vertices that are not delayed.** This type has at most one part $P^{(a)}$, which consists of all non-cycle vertices in Ψ^* that are not in $\mathtt{D}(\Psi^*)$.

Type b: **One cycle vertex with its delayed set.** For each cycle vertex $\psi_t^* \in \Psi^*$ that is the only vertex in the cycle protected by Ψ^*, ψ_t^* and the set of vertices protected by Ψ^* and delayed by ψ_t^* form a part. That is, given such a vertex ψ_t^*, they form a Type-b part $P_t^{(b)} = \{\psi_t^*\} \cup \mathtt{D}_{\Psi^*}(\psi_t^*)$. Note that $P_t^{(b)}$ might be empty for some t. We call t the *representative vertex* of $P_t^{(b)}$.

Type c: **Two cycle vertices with their delayed set.** For each cycle C that contains two vertices ψ_t^* and ψ_s^*, and at least one of ψ_t^* and ψ_s^* is not delayed by Ψ^*, these two vertices and their delayed sets form a Type-c part $P_{t,s}^{(c)}$. That is, $P_{t,s}^{(c)} = \{\psi_t^*, \psi_s^*\} \cup \mathtt{D}_{\Psi^*}(\psi_t^*) \cup \mathtt{D}_{\Psi^*}(\psi_s^*)$. For some pair of t and s, $P_{t,s}^{(c)}$ might be empty. We call ψ_t^* and ψ_s^* the *representative vertices* of $P_{t,s}^{(c)}$.

A vertex might be eligible for both parts of Type-b and Type-c. In such cases, we give priority to assign it to a Type-c part. If a vertex $\psi_t^* \in \Psi^*$ belongs to at least two cycles, observe that the fire can approach ψ_t^* from only one of these cycles. This is the only cycle containing ψ_t^* where two firefighters might be assigned. Protecting ψ_t^* saves all other cycles entirely. Hence, this cycle determines whether ψ_t^* is assigned to a Type-b or a Type-c part.

▶ **Lemma 10.** *The partition $\mathcal{P}^*$ is a proper, cycle-respecting partition of Ψ^*.*

Charging function

We define the charging function $\chi : \mathcal{P}^* \to 2^{\Psi^{\mathrm{A}}}$ according to the type of parts.

Type-a part. $\chi(P^{(a)})$ consists of the vertices $\psi_t^{\mathtt{A}}$ corresponding to all $\psi_t^* \in P^{(a)}$. Moreover, if $\psi_t^{\mathtt{A}}$ is a cycle vertex, then all of its peers on the same cycle are also included.

Type-b and Type-c parts. $\chi(P)$ consists of all $\psi_t^{\mathtt{A}}$ corresponding to every $\psi_t^* \in P$, together with their peers (if any). Additionally, for every $\psi_t^* \in P$, we further include:

- `Break-cycle rule`: If $\psi_t^{\mathtt{A}}$ is protected via ImprovedBreak, we also include all vertices covered by C, where C is the cycle containing $\psi_t^{\mathtt{A}}$.
- `Cool-down rule`: If $\psi_t^{\mathtt{A}}$ protects the heaviest vertex at time t due to a positive `Cool-down` timer, which was set in the previous break-cycle at time $s < t$, we also include $\psi_s^{\mathtt{A}}$ and all vertices covered by C_s containing $\psi_s^{\mathtt{A}}$.

By a rough approximation based on how the charged subset of Ψ^{A} is defined, each vertex in Ψ^{A} can only be charged by at most 13 parts. By a more careful counting, we get:

▶ **Observation 11.** *Any vertex in Ψ^A is charged at most 5 times.*

Next, we analyze the exclusive weight of a part by its type. Starting with Type-a parts.

▶ **Lemma 12** (Non-cycle vertices in Ψ^* that are not delayed)**.** *If $P^{(a)}$ is the Type-a part, then $w'(P^{(a)}) \leq \sqrt{n} \cdot w(\chi(P^{(a)}))$.*

▶ **Lemma 13** (One cycle vertex with its delayed set)**.** *If $P_t^{(b)}$ is a Type-b part, then $w'(P_t^{(b)}) \leq 4\sqrt{n} \cdot w(\chi(P_t^{(b)}))$.*

▶ **Lemma 14** (Two cycle vertices with their delayed set)**.** *If $P_{t,s}^{(c)}$ is a Type-c part, then $w'(P_{t,s}^{(c)}) \leq 4\sqrt{n} \cdot w(\chi(P_{t,s}^{(c)}))$.*

By combining Lemmas 12, 13, and 14, together with Observation 11, we can conclude that algorithm $\mathrm{ALG}_{\mathsf{C}}$ is $\Theta(\sqrt{n})$-competitive.

▶ **Theorem 15.** *There is a $(20\sqrt{n}+1)$-competitive algorithm ALG_{C} for the online firefighting game on cactus graphs, which is asymptotically optimal.*

5 Cactus graphs with even firefighters each round

In this section, we extend our framework to a special case of online firefighting on cactus graphs, where, in each round, either there is no firefighter available or an even number of firefighters are available. Similar to the 2-competitive greedy algorithm for trees [3], the availability of an even number of firefighters each round allows the algorithm to always greedily protect the heaviest component, even when this component is a cycle. We show that a greedy algorithm modified from our algorithm for cactus graphs is exactly 3-competitive.

5.1 The greedy algorithm $\mathrm{ALG}_{\mathtt{E}}$

We follow the graph reduction framework. In each round, the algorithm $\mathrm{ALG}_{\mathtt{E}}$ places the firefighters in a way that maximizes the number of vertices immediately saved during this round. More specifically, when two or more firefighters are available, $\mathrm{ALG}_{\mathtt{E}}$ protects the heaviest cycle $\widetilde{C}_1$ if it is heavier than the total weight of the two heaviest vertices v_1 and v_2 in the graph. Otherwise, the algorithm protects the heaviest vertex v_1. When there is only one firefighter available, the algorithm protects v_1. Note that although each round an even number of firefighters become available, due to its greedy nature, our algorithm might choose to protect v_1 first, and the algorithm might be left with an odd number of firefighters.

5.2 Competitive analysis

Given the sets of vertices Ψ^{A} and Ψ^* protected by $\mathrm{ALG}_{\mathtt{E}}$ and by a non-redundant offline optimal solution, respectively, we follow the analysis framework introduced in Section 2. Note that under algorithm $\mathrm{ALG}_{\mathtt{E}}$, a cycle can be protected by vertices $\psi_t^{\mathtt{A}}$ and $\psi_{t+1}^{\mathtt{A}}$ in Ψ^{A}, with t being either odd or even.

Partition $\mathcal{P}^*$ of Ψ^*

We partition the vertices in Ψ^* into three types of parts, which is an adaption from the partition for analyzing $\mathrm{ALG}_{\mathtt{C}}$:

Type a: **One non-cycle vertex that is not delayed.** For any non-cycle vertex $\psi_t^* \in \Psi^*$ that is not delayed by Ψ^*, there is a part $P_t^{(a)} = \{\psi_t^*\}$, which is a singleton, and ψ_t^* is the representative vertex in $P_t^{(a)}$.

Type b: **One cycle vertex that is not delayed.** For each cycle vertex $\psi_t^* \in \Psi^*$ that is the only vertex in the cycle protected by Ψ^*, there is a Type-b part $P_t^{(b)} = \{\psi_t^*\} \cup \mathtt{D}_{\Psi^*}(\psi_t^*)$. We call ψ_t^* the *representative vertex* of $P_t^{(b)}$.

Type c: **Two cycle vertices with their delayed set.** For each cycle C that contains two vertices ψ_t^* and ψ_s^*, and at least one of ψ_t^* and ψ_s^* is not delayed by Ψ^*, there is a part $P_{t,s}^{(c)} = \{\psi_t^*, \psi_s^*\} \cup \mathtt{D}_{\Psi^*}(\psi_t^*) \cup \mathtt{D}_{\Psi^*}(\psi_s^*)$. Recall that the index of $\psi^*_{\min\{t,s\}}$ is the time label in our framework. We call $\psi^*_{\min\{t,s\}}$ the *representative vertex* of $P_{t,s}^{(c)}$.

Charging function

Given any part P, our charging function maps every representative vertex ψ_t^* to $\psi_x^{\mathtt{A}}$ and $\psi_{x+1}^{\mathtt{A}}$, where x is the *anchor* of P, defined as the largest odd number which is at most t (where t is the first vertex in P protected by OPT).

▶ **Theorem 16.** *There is a 3-competitive algorithm ALG_E for the online firefighting game on cactus graphs with even firefighters in each round.*

Proof sketch. We prove the theorem by first showing that, given a part P with anchor time x, $w'(P) \leq w_x(v_1)$ or $w'(P) \leq w_x(\widetilde{C}_1)$. Then, by case distinction whether $\psi^{\mathtt{A}}_x$ and $\psi^{\mathtt{A}}_{x+1}$ are protecting the heaviest vertex or protecting the heaviest cycle (perhaps with another vertex $\psi^{\mathtt{A}}_{x-1}$ or $\psi^{\mathtt{A}}_{x+1}$), we show that, given any part P with anchor time x, $w_x(\widetilde{C}_1) \leq \bar{w}(\chi(P))$ and $w_x(v_1) \leq \bar{w}(\chi(P))$. Thus, we can conclude that for any part P with anchor time x, $w'(P) \leq \bar{w}(\chi(P))$ since $\chi(P) = \{\psi^{\mathtt{A}}_x, \psi^{\mathtt{A}}_{x+1}\}$.

Consequently, since the charging function χ charges each part P to $\psi^{\mathtt{A}}_x$ and $\psi^{\mathtt{A}}_{x+1}$ where x is the anchor time of P, each $\psi^{\mathtt{A}}_t$ can be charged at most twice. Hence, by our framework, $\frac{\mathrm{OPT}(\mathcal{I})}{\mathrm{ALG}_{\mathtt{E}}(\mathcal{I})} = \frac{w(\Psi^*)}{w(\Psi^{\mathrm{A}})} = \frac{w'(\Psi^*)}{w(\Psi^{\mathrm{A}})} + 1 = \frac{\sum_P w'(P)}{w(\Psi^{\mathrm{A}})} + 1 \leq \frac{\sum_P w(\chi(P))}{w(\Psi^{\mathrm{A}})} + 1 \leq \frac{2w(\Psi^{\mathrm{A}})}{w(\Psi^{\mathrm{A}})} + 1 = 3.$ ◀

By showing an instance on which $\mathrm{ALG}_{\mathtt{E}}$ saves only a third of the vertices that the offline optimal offline algorithm is able to save, we can conclude that the analysis of $\mathrm{ALG}_{\mathtt{E}}$ is tight.

▶ **Theorem 17.** *The algorithm ALG_E is 3-competitive and the analysis is tight.*

Proof sketch. Let G be a graph with two cycles with 8 vertices, $C_x = (r, x_1, x_2, \cdots, x_7)$ and $C_y = (r, y_1, y_2, \cdots, y_7)$. Each vertex x_1, x_2, y_1, and y_2 has β degree-1 vertices adjacent to it. There are two other paths $(r, a_1, a_2, \cdots, a_{\beta+6})$ and $(r, b_1, b_2, \cdots, b_{\beta+6})$. The firefighter sequence is $(2, 0, 0, 0, 4, 0, \cdots)$. That is, $f_1 = 2$, $f_5 = 4$, and $f_i = 0$ for all $i \notin \{1, 5\}$. ◀

References

1 David Adjiashvili, Andrea Baggio, and Rico Zenklusen. Firefighting on trees beyond integrality gaps. *ACM Trans. Algorithms*, 15(2):20:1–20:33, 2019. doi:10.1145/3173046.

2 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

3 Pierre Coupechoux, Marc Demange, David Ellison, and Bertrand Jouve. Firefighting on trees. *Theor. Comput. Sci.*, 794:69–84, 2019. doi:10.1016/J.TCS.2019.01.040.

4 Marc Demange, David Ellison, and Raffaella Gentilini. Online firefighting on grids. In Alessandra Cherubini, Nicoletta Sabadini, and Simone Tini, editors, *Proceedings of the 20th Italian Conference on Theoretical Computer Science, ICTCS 2019, Como, Italy, September 9-11, 2019*, volume 2504 of *CEUR Workshop Proceedings*, pages 91–96. CEUR-WS.org, 2019. URL: https://ceur-ws.org/Vol-2504/paper11.pdf.

5 Stephen Finbow, Andrew D. King, Gary MacGillivray, and Romeo Rizzi. The firefighter problem for graphs of maximum degree three. *Discret. Math.*, 307(16):2094–2105, 2007. doi:10.1016/J.DISC.2005.12.053.

6 Bert Hartnell. Firefighter! an application of domination. In *the 24th Manitoba Conference on Combinatorial Mathematics and Computing, University of Minitoba, Winnipeg, Cadada, 1995*, 1995.

7 Bert Hartnell. Firefighting on trees: how bad is the greedy algorithm? *Congressus Numerantium*, 145:187–192, 2000.

8 Gary MacGillivray and Ping Wang. On the firefighter problem. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 47:83–96, 2003.