Theoretical Aspects of Intruder Search

MA-INF 1318 Manuscript Wintersemsester 2015/2016

Elmar Langetepe

Bonn, 19. October 2015

The manuscript will be successively extended during the lecture in the Wintersemester. Hints and comments for improvements can be given to Elmar Langetepe by E-Mail `elmar.langetepe@informatik.uni-bonn.de`. Thanks in advance!

II

Contents

Chapter 1

Introduction

In this lecture we consider intruder and evader problems from an algorithmic point of view. We discuss problems in the context of motion planning in discrete and continuous environments. Discrete environments will be mainly defined by graphs whereras in the continuous case we will make use of the Euclidean plane or subsets of the Euclidean plane.

Generally, there is a set of searcher (or guards) and another set of intruder (or evaders) that compete with each other. The intruder tries to escape from the searcher in the given environment. The other way round the guards would like to protect a certain area from the intruder. Intruder and searcher can have different facilities or sensors. There is always some kind of dynamics, the searcher and/or the intruder can move or perform tasks over time.

The intruder search problem might also be considered as a scenario for the cleaning or the enclosure of a spreading contamination. For any moment in time the current contamination represents all possible remaining positions of the intruder. So the task of the searcher is to reduce the number or to bound the area of these positions and to give a guarantee where the intruder is located.

More precisely, from a theoretical point of view we would like to answer the following questions:

- Computational complexity (i.e. existence of a strategy, question for the number of guards required, running time for the computation of a strategy)

- Correctness or Failure (i.e. success of a given strategy, limits of the strategy)

- Efficiency (i.e. number of steps or path length required for a given strategy, area saved by the guards)

- Optimality (i.e. cost measures for the efficiency and its analysis)

We will also see that there are many interesting open questions. Sometimes simulations will be helpful for getting more insight into the problem and a conjecture for a solution. Additionally, the usage of simulation tools will be part of this lecture.

1.1 Introductory examples

In the following we would like to discuss some simple introductory examples and relate it to the above questions. This gives some insight into the nature of the problems we would like to discuss and also shows the variety of methods that will be applied.
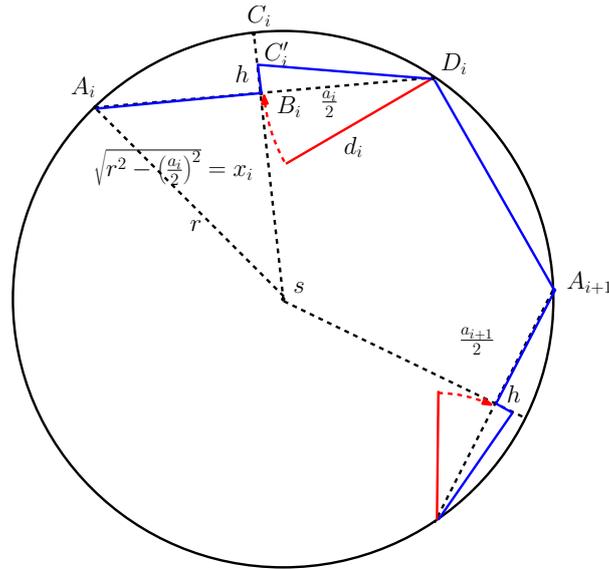
Figure 1.1: Constructing a polygon with doors for a set of n integer $a_1, a_2, \ldots, a_n$. The radius r and the speed v are chosen appropriately so that after $t + 0.5$ time steps the intruders can reach any point C_i' but before t time steps no point B_i can be reached.

1.1.1   Protecting parts of a polygonal area from a set of intruders

Let us assume that we have a simple room modeled by a polygonal area P (a simple polygon) and there is a set of n doors that can be closed successively. Additionally, there is a set of intruders starting at some point s. The intruder moves with speed v. For a given set of n doors each door can protect the area behind the door. The time for closing a single door d_i is given by a value a_i. Any door d_i protects a portion A_i of the polygon from the intruders. (Somehow it makes sense that the time a_i for closing the door is proportional to the area A_i that can be saved by d_i.) We would like to close as many doors as possible, thus maximizing the portion that cannot be entered by the intruders. The doors can only be closed one after the other.

The question is, which doors should be closed successively, so that the intruders cannot enter the part of the polygon behind the doors. We would like to maximize the area that will be safe. As a matter of computational complexity we show that this problem is NP-hard.

Optimal-Closing-Sequence
Instance: A simple polygon, a set of n intruders with starting positions and speed 1, a set of m doors that can be closed successively in time c_i and saves area A_i.
Output: Compute the optimal sequence of doors that has to be closed for maximizing the area saved.

Theorem 1 *Computing the optimal closing-sequence for a set of doors in a polygon is NP-hard in general.*

Proof. The subset-sum problem can be reduced to our problem. We have n integer numbers $a_1, a_2, \ldots, a_n$ and a integer threshold t and we are searching for a subset-sum, so that the sum does not exceed t but is as close as possible to t. [1] The subset-sum problem will be transformed into our problem as depicted in Figure 1.1. We make use of a circle C_r of radius r. Let s be the center of the circle. Along the boundary we use non-intersecting secants of length a_i with endpoints A_i and D_i in clockwise order. A ray starting at s and running through the midpoint

---

[1] Note that partition can be reduced to the subset-sum problem.

B_i of each secant hits the boundary of C_r at some point C_i. Along the segment B_iC_i we use a point C_i' a fixed distance h away from B_i. For the construction of the polygon we use the chain A_i,B_i,C_i',D_i and connect D_i with A_{i+1} and also finally D_n with A_1. There is a door d_i for each pair of points B_i and D_i. This door can be closed within a_i time steps and saves a region of area $\frac{a_i \cdot h}{4}$ proportional to a_i.

Now we choose the speed v so that $v(t+0.5) = r$ holds. This means that after $t+0.5$ time steps an intruder can reach any C_i'. Additionally, we have to take care that after t time steps, none of the entry points B_i closest to s can be reached by an intruder. Therefore, we choose r large enough so that $vt < x_i = \sqrt{r^2 - \left(\frac{a_i}{2}\right)^2}$ holds. Substituting v by $\frac{r}{(t+0.5)}$ gives

$$\left(\frac{a_i}{2}\right)^2 < \left(1 - \frac{t^2}{(t+0.5)^2}\right) r^2$$

which can be fulfilled for any a_i.

Altogether, after $t+0.5$ time steps any entrance point B_i could be passed but after t time step none of the entrance points have been visited. If we would like to save the maximum area from the intruders, we have to compute a maximum sum of values a_i not exceeding $t+0.5$ but beeing as close as possible to t. This gives a solution for the subset-sum problem.

Conversely, for a solution of the subset-sum problem we obtain the maximum sum of values a_i smaller than t, thus we save a maximum area if we close the corresponding doors successively.

□

Exercise 1 *Make use of the* `GeoGebra`*-tool and compute an interactive construction of the above reduction for a finite set of elements $a_1, a_2, \ldots, a_n$. Give examples $a_1, a_2, \ldots, a_n$ and t where the subset-sum question is answered with Yes and with No, respectively.*

1.1.2 Catching an evader in a grid world

The next example considers a discrete problem variant and a different intention.

There is some evader that can move in a rectangular grid world and a guard would like to enclose and catch the evader. For the enclosement and for catching the evader the guard can block and check cells of the grid while the evader is moving. The evader starts at a single cell and can visit the cells of the 4Neighborship in one step or could stay in its current cell. Simultaneously, the guard can block or check k of the cells. Any cell that was blocked cannot be entered by the evader anymore. In order to synchronize the movements we first let the evader move and then the guard can choose its k blocking and checking cells; see Figure 1.2.

We do not have an idea of the strategy of the evader. Therefore, simply all possible locations of the single evader will be considered and marked by red circles. The blocked cells and checked cells are marked by black circles. We would like to find an efficient strategy for the guard. That is, we would like to enclose the evader as fast as possible. After that the guard simply checks all the remaining red cells. The game ends when there is no single red cell any more.

Evader-Enclosement (Grid-Graph):
Instance: A rectangular grid, a start vertex s of the evader and k protecting guards per time step.
Output: Compute a protection strategy that encloses the evader (and finally find the evader).

We first discuss failure and correctness for different values of k, we will see that some machinery is necessary.

Lemma 2 *Catching an evader in a grid world by setting $k = 1$ blocking cells after each movement of the evader cannot succeed in general.*
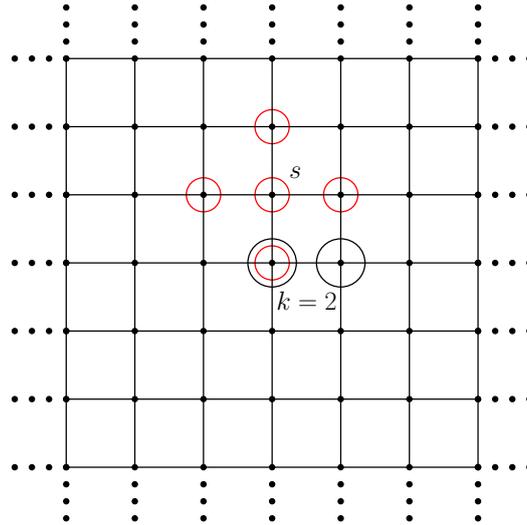
Figure 1.2: The evader starts at s. In the first step he can move to the adjacent vertices of the grid. For $k = 2$ the guard blocks and checks two cells afterwards. Now the evader can move on.

Proof. We show that the evader always has an escape strategy. Or the other way round, a guard can never cancel out all cells marked red.

For a formal proof we show that it is already impossible to catch the evader in a single quadrant of cells. The evader starts at the source of the quadrant, also called root in the following; see Figure 1.3. For any step $l \geq 1$ let D_l denote the set of vertices that are distance l away from the root vertex. This means, that these cells lie on the l-th diagonal of the quadrant.

Let r_l denote the number of blocked cells in D_{l+1}, D_{l+2},... at the end of the l-th step. The blocked cells in D_{l+1}, D_{l+2},... can be considered to build a reserve for the future. Let $B_l \subseteq D_l$ denote the number of red cells at the end of the l-th step distance l away from the root. We have $r_0 = 0$.

By induction we show that $B_l \geq 1 + r_l$ holds for all $l \geq 1$. In the first step $l = 1$ the evader can move to both vertices of D_1 and the guard can block a single cell, either in D_t for $t = 0, 1$ or in D_t for $t > 1$. The latter one means $r_1 = 1$. In any case we have $B_1 \geq 1 + r_1$.

Let us assume that the statement holds after $l \geq 1$ steps. We can assume $B_l \geq 1 + r_l$. This means that in step l the evader can visit $r_l + 1 + 1$ cells in diagonal D_{l+1}. But some of them might already be blocked by the reserve and another one can be blocked by the guard in this step. Let us assume that there are $x \leq r_l$ blocked cells in D_{l+1}.

This means that at first (the evader moves) we have $B_{l+1} \geq 1 + 1 + r_l - x$ red cells in D_{l+1}. Now the guard can block another cell in step l also. Either in D_t for $t \leq l$, in D_t for $t = l + 1$, or in D_t for $t > l + 1$.

In the latter case we have $r_{l+1} = 1 + r_l - x$ and in the former cases we have $r_{l+1} = r_l - x$. Now the guard can only reduce B_{l+1} by blocking a vertex in D_{l+1}. In any case we have $B_{l+1} \geq 1 + r_{l+1}$ which gives the conclusion.   □

Note that for the pure enclosement of the evader it seems reasonable that in step l the guard should never block a cell that was marked by a red circle in one of the previous rounds $< l$.

Exercise 2 *Give a formal argument for the fact that a fast guard strategy should first enclose the evader and then check the remaining cells. This means that in step l the guard strategy will not choose a red marked vertex that was marked in a step $< l$ until the evader cannot move anymore.*
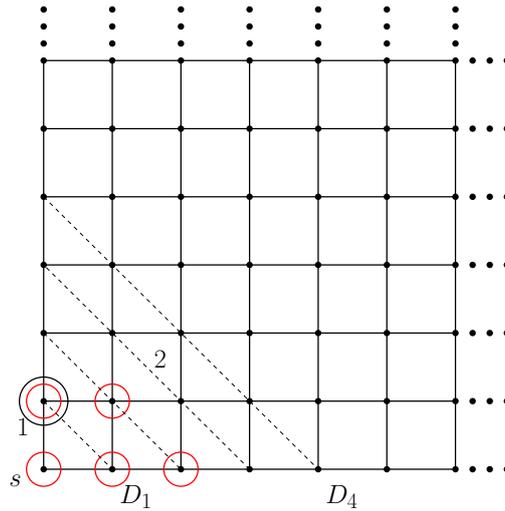
Figure 1.3: The situation for $k = 1$ after step 2. We have $r_2 = 1$ and $B_2 = 1 + r_2 = 2$.
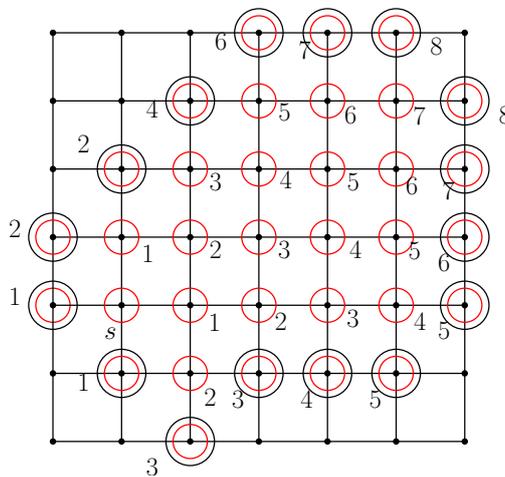
Figure 1.4: For $k = 2$ there is a guard strategy that encloses the evader after 8 steps. Now the evader can be catched by checking the remaining 18 cells.

Fortunately for $k = 2$ there is a successful strategy. We can just give the sequence of steps for the enclosement as presented in Figure 1.4. After that the remaining cells can be cleaned.

Lemma 3 *There is a successful guard strategy for $k = 2$. The evader can be enclosed after 8 steps. With 9 additional steps the evader will be catched.*

Proof. By construction; see Figure 1.4. □

The above situation can also be considered as the problem of containing a spreading fire. We slightly change the rules of the game. Let us assume that the firemen first blocks k cells and then the fire can spread from the source s to its neighboring cells. A burning cell can never be blocked any more.

In this sense and for $k = 2$ a firemen can enclose a spreading fire in 8 steps as depicted in Figure 1.4. Here only 18 vertices get burned. We give a formal proof that this is optimal.

Lemma 4 *For the outbreak of a fire on a single source in a grid and the usage of two firefighters*

per time step any successul strategy encloses an area of at least 18 burning vertices. This bound is tight.

Proof. For the proof of the statement we make use of an Integer Linear Program as follows. We already know that it is possible to enclose 18 burning cells by the strategy given in Figure 1.4 in 8 time steps. Therefore we make use of a fixed constant $T > 8$ for the number of time steps. Additionally, we use a grid of size $l \times l$ for another fixed constant l.

Let $L = \{(x, y) | |x| \leq l \text{ and } |y| \leq l\}$ and $0 \leq t \leq T$. For $v \in L$ let $N(v)$ denote its 4Neighborship in L. For the Integer Program we use the following variables for $v \in L$.

$$b_{v,t} = \begin{cases} 1 & : & \text{vertex } v \in L \text{ burns before or at time } t \\ 0 & : & \text{otherwise} \end{cases}$$

$$d_{v,t} = \begin{cases} 1 & : & \text{vertex } v \in L \text{ is defended before or at time } t \\ 0 & : & \text{otherwise} \end{cases}$$

We would like to minimize the number of vertices that become burned.

$$
\begin{array}{llll}
\text{Minimize} & \sum_{v \in L} b_{v,T} \\
\text{so that} & b_{v,t} + d_{v,t} - b_{w,t-1} & \geq & 0 & : & \forall v \in L, v \in N(w), 1 \leq t \leq T \\
& b_{v,t} + d_{v,t} & \leq & 1 & : & \forall v \in L, 1 \leq t \leq T \\
& b_{v,t} - b_{v,t-1} & \geq & 0 & : & \forall v \in L, 1 \leq t \leq T \\
& d_{v,t} - d_{v,t-1} & \geq & 0 & : & \forall v \in L, 1 \leq t \leq T \\
& \sum_{v \in L}(d_{v,t} - d_{v,t-1}) & \geq & 2 & : & \forall 1 \leq t \leq T \\
& b_{v,0} & = & 1 & : & v \in L \text{ is the origin } (0,0) \\
& b_{v,0} & = & 0 & : & v \in L \text{ is not the origin } (0,0) \\
& d_{v,0} & = & 0 & : & \forall v \in L \\
& d_{v,t}, b_{v,t} & \in & \{0,1\} & : & \forall v \in L, 1 \leq t \leq T
\end{array}
$$

The first inequality triggers the spread of the fire. The neighbor of a burning vertex is either blocked or it burns in the next step. Note that with this condition any vertex can also burn accidently. But this will not happen in the optimal solution. The next inequality prevents a defended vertex from burning and a burning vertex from beeing defended. The next two inequalities take care that the status of the vertices remains the same after burning or beeing defended. The sum condition refers to the number $k = 2$ in any step. Initially only one vertex is burning and no vertex is defended. Finally, the program gets binary.

The corresponding program was run for example by Develin and Hartke [] and for $l = 6$ and $T = 9$ and the origin $(0, 0)$ the solution of Figure 1.4 was achieved.   □

The above proof is an example where we make use of a computer aided process for the conclusion of a theoretical proof. With similar arguments it is also possible to show that 18 vertices cannot be enclosed within less than 8 steps.

1.1.3   Enclosing a fire by a single circle

In the last section we have seen that catching an evader and the enclosement of a fire are very similar problems which can be formulated in a similar model. Let us consider the fire containment problem in the following continuous setting.

The fire is already burning for a while and the fire spreads into any direction with the same unit speed 1. There is a single firefighter which can build a firebreak with speed $v > 1$. Geometrically
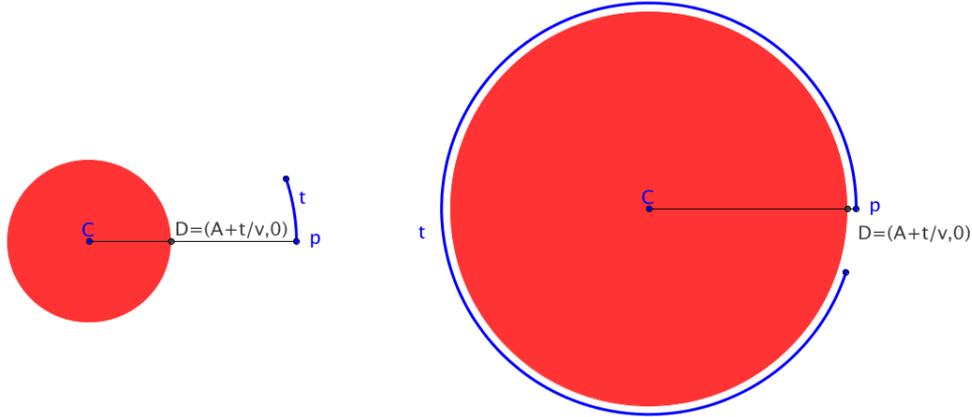
Figure 1.5: The firefighter builds a firebreak of length t with speed $v > 1$ and the fire expands from the source with speed 1. The circular strategy is successful here.

we can assume that in the beginning of the game the fire is a circle of radius A from a source C and the circle grows with speed 1 while the firefighter builds a path t with speed $v > 1$; see Figure 1.5. We call this the *geometric firefighter problem*.

Geometric Firefigther Problem
Instance: A circle with center C of radius A that grows with unit speed. An agent who builds a firebreak with speed $v > 1$
Output: Compute a strategy that finally fully enclose the spreading fire.

A simple question is: How fast must the firefighter be in order to enclose the fire by a single circular loop around the source of the fire? Let us assume that we can choose an arbitrary starting point p for the firefigther.

Lemma 5 *Enclosing a fire of extension A with a single circular loop around the source of the fire is possible, if and only if the speed v of the firefigther is larger than 2π.*

Proof. We assume that the fire source is given by the origin and we choose a point $p = (A+x, 0)$ away from the fire. A single circular loop around the origin has length $2\pi(A + x)$ and takes $\frac{2\pi(A+x)}{v}$ time. We can enclose the expanding circle if this is smaller than or equal to x. This gives $\frac{2\pi A}{x} + 2\pi \leq v$. For $v > 2\pi$ there will always be an x so that the inequality holds. For $v \leq 2\pi$ we can never fulfill the required inequality.   □

Exercise 3 *Consider the above circular strategy. Assume that it is allowed to move along an arbitrary circle (around an arbitrary center). Is it possible to improve the above bound? I.e., is the above bound tight for arbitrary circles or can we always succeed with a circle and speed less than 2π?*

1.1.4 Simulation and conjecture for a discrete spiral strategy

Finally, we would like to motivate that in some settings it is helpful to make some simulation. We would like to enclose a spreading contamination in a grid world and apply a strategy that always moves close to the boundary of the contamination. The agent has to move from one cell to the other and can build a wall as in the previous setting. The agent either moves clockwise or counterclockwise around the starting contamination. We assume that the contamination is a grid square in the beginning.
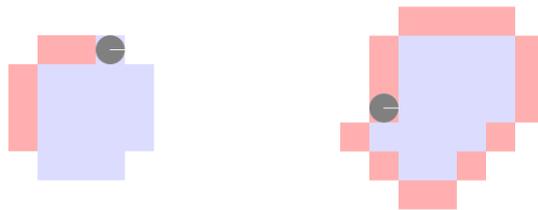
Figure 1.6: The fire spreads after 30 steps and the agent build a wall cell within 5 time steps. The lefthandside figure shows the situation after 30 time steps. The contaminations is enclosed in a single loop after a few more steps.
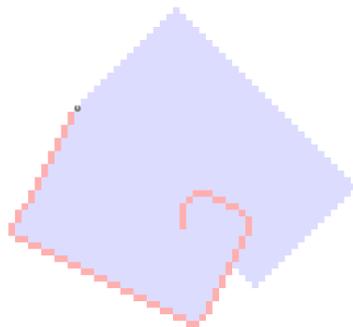
Figure 1.7: The fire spreads after 30 steps and the agent build a wall cell within 15 time steps. After a while it seems that the spiral strategy will not succeed.

Parameters: Moving from one cell to the other takes one time step. Building a wall element and cleaning a cell takes b time steps. After n time steps the contamination *spreads*. This means that the 4Neighborship of any contamintated cell becomes contaminated but cells that contain a wall element cannot be contaminated anymore.

Discrete Firefigther Problem
Instance: A grid contamination some size B that spreads to its 4Neighborship after n time steps. An agent who cleans a cell, builds a wall and leaves the celle within b time steps.
Output: Compute a strategy that finally fully enclose the spreading fire.

Figure 1.6 shows an example for a 3×3 starting contamination with a single agent, $b = 5$ and $n = 30$ after exactly 30 time steps and when the task is finished. Figure 1.7 shows the case where $n = 30$ and $b = 15$, the spiral strategy does not succeed.

Experiments from the thesis of Smiegilski [] give the following conjecture.

Conjecture 1 *For a grid fire thats spreads after n time steps and an agent that builds a wall within b time steps, the spiral strategy only succeeds if $b < \frac{n-1}{2}$ holds.*

Chapter 2

Discrete Scenarios for Contaminations

In this chapter we would like to discuss several results on the firefigther or intruder problem on discrete graph structures. The main problem is NP-complete already for trees but there are some variants that can be solved in polynomial time. Besides there are efficient approximation schemes.

We consider dynamic problems for connected graphs $G = (V, E)$ with a root vertex r where the fire or contamination starts only at r. At each time step the firefigther can protect p vertices that are not contaminated. Afterwards the contamination spreads to all unprotected vertices neighboring a contaminated vertex. A protection strategy describes the sequence S of vertices that will be protected over time.

Firefighter Decision Problem for Graphs (Saving k vertices):
Instance: A Graph $G = (V, E)$ of degree d with root vertex r and p guards per step and an integer k.
Question: If the fire breaks out at vertex r, is there a protecting strategy so that at least k vertices can be protected?

An optimal protection strategy protects the maximum number of vertices. The problem is NP-complete already for trees of small degree. This general proof is a bit technical and would take some time so we will use a somewhat simpler proof for NP-completeness on graphs.

2.1 Graphs

2.1.1 Polynomial time algorithm for special graphs

Let us assume that the degree d of the vertices of G is bounded to 3. Thus it only makes sense to set p to one. Otherwise the game ends in the very beginning. If the start vertex has degree 2, there is a simple polynomial time algorithm. In principle we force the fire to spread along a path until finally some vertex is enclosed. The running time depends on the nature of this vertex.

We first introduce this measure and a corresponding strategy for all vertices u of G. Let $\mathrm{dist}(u, r)$ denote the length of a shortest path from r to u. Let V_1 denote all vertices of degree 1 and let V_2 denote all vertices of degree 2 and let V_c denote all vertices of degree 3 that belong to a cycle. Let $C(u)$ denote the smallest cycle containing u. The following Lemma shows how long it could take to enclose a vertex u which is finally on fire; see also Figure 2.1.
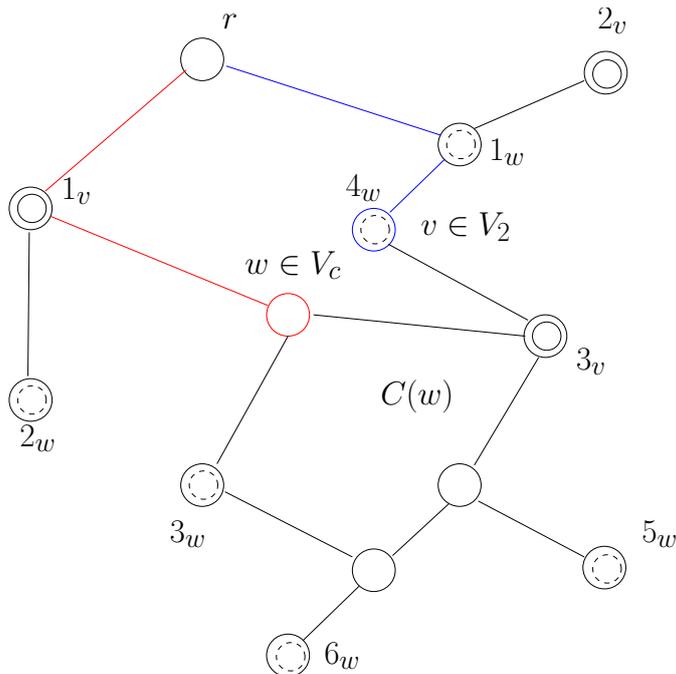
Figure 2.1: Protecting a vertex $v \in V_2$ and $w \in V_c$. The blue and red path denotes the shortest path. The labels i_v and j_w denote the vertices that will be protected at step i and j until v and w are finally enclosed, respectively.

Lemma 6 *A vertex $u \in V_1 \cup V_2$ can be enclosed in time $dist(u, r) + 1$ and only $dist(u, r) + 1$ vertices are on fire. A vertex $u \in V_c$ can be enclosed in time $dist(u, r) + C(u) - 1$ and only $dist(u, r) + C(u) - 1$ vertices are on fire.*

Proof. For the above statements we simply follow a shortest path from r to u. In each time step we defend the vertex that is adjacent to a burning vertex but not on the path to u. This shows the property for all $u \in V_1 \cup V_2$.

For $u \in V_c$ after $dist(r, u) + 1$ time steps we are reaching u. For the remaining steps we protect the vertex not on $C(u)$ but with a burning neighbor. Thus we encircle u in any case.   □

Now we would like to prove a structural property for an optimal strategy. The first protected vertex can always be one of the neighbors of r and after that it also suffices to always defend the vertices neighboring the fire as the following Lemma shows.

Lemma 7 *For a setting $(G, r, 1)$ where G has maximal degree 3 and root r has degree ≤ 2 there is always an optimal protection strategy that protects the neighbor of a contaminated vertex in each time step.*

Proof. If r has degree 1, the statement is trivial. So let r be of degree 2, assume that the statement is false and let $(G, r, 1)$ be a minimal counterexample. Let v_1 and v_2 be the neighbors of r. If there is an optimal strategy that first protects the neighboring vertex v_1 of r, let G' denote the Graph that is attained by deleting r and v_1. Thus $(G', v_2, 1)$ is also a counterexample that contradicts the minimality.

So we can assume that in a minimal counterexample in the first step neither v_1 nor v_2 will be protected. Let u be the vertex protected first with shortest path distance ≥ 2 to r.

Consider the end of the optimal strategy. If u has no burning neighbor, the strategy could not have been optimal. If u has only one burning neighbor, we can improve the strategy by

protecting this neighbor in the first step. If there are two neighbors of u that are burning at the end of the strategy, the vertex u lies on a cycle which is completely burned except for u. Thus the strategy implied in Lemma 6 for one of the neighbors of u is an optimal variant. $\quad\square$

Exercise 4 *Show that the statement of Lemma 6 does not hold for a degree 3 starting vertex. Present a non-trivial example for the optimal strategy of a graph G of degree 3 and root vertex of degree 2 based upon the above calculations.*

Finally, we suggest the following strategy. Let

$$
f(u) := \begin{cases} \operatorname{dist}(u,r) + 1 & : \quad \text{if } u \in V_1 \cup V_2 \\ \operatorname{dist}(u,r) + C(u) - 1 & : \quad \text{if } u \in V_c \setminus V_2 \\ \infty & : \quad \text{otherwise} \end{cases}
$$

and find a vertex u with $f(u) = \min_{x \in V} f(x)$. Enclose this vertex by the path strategy.

Theorem 8 *For a problem instance $(G, r, 1)$ of a graph G of maximum degree 3 and a root vertex of degree 2 the above strategy is optimal.*

Proof. In Lemma 7 we have shown that there is always an optimal strategy that protects the neighbor of a contaminated vertex. Let u be one of the vertices burning last. If u has degree 1 or 2, we require at least $d(r, u) + 1$ time steps and $d(r, u) + 1$ are burning, which is optimal.

If u has degree one there are three neighbors n_1, n_2 and n_3. If only one neighbor n_1 is on fire, u, n_2 and n_3 lie on a cycle that is totally burning, a contradiction to the path strategy.

So let us assume that two neighbors n_2 and n_3 of u are protected and one neighbor n_1 is on fire. There should be another neighbor of either n_2 or n_3 that is on fire. If this is not the case, we could have blocked u one step earlier. This means that one of the neighbors n_2 or n_3 of u has to build a cycle $C(u)$ with u and n_1 so that any vertex of this cycle burns except for u. In this case the above strategy optimizes the best such cycle and is as least as good as the optimum.

Since the degree is bounded, there are no other cases. $\quad\square$

Theorem 9 *For a problem instance $(G, r, 1)$ of a graph $G = (V, E)$ of maximum degree 3 and a root vertex of degree 2 the decision problem can be solved in polynomial time and the maximum number of vertices that can be saved is $|V| - \min_{x \in V} f(x)$.*

Proof. By the above considerations. Just compute $f(x)$ for every vertex $x \in V$ taking the shortest path and the smallest cycle into account. $\quad\square$

Exercise 5 *Analyse the precise running time for the computation of the optimal strategy. That is, present an algorithm and its running time that computes $f(x)$ for every vertex $x \in v$ efficiently.*

Lemma 6 suggests to always choose a vertex close to the fire. For an arbitrary given tree with root r it seems that this is also the best option we will prove this statement in Lemma 12. It seems that for trees a greedy approach could be optimal but this is not the case as Section 2.2.1 shows.

2.1.2 NP-Completeness for graphs

Theorem 10 *The firefighter decision problem for graphs is NP-hard.*

Proof. We reduce the k-Clique problem to the firefigther problem. So let $G = (V, E)$ be a graph and k an integer: Is there a Clique of size k inside G? This question is known to be NP-complete. We assume that G has at least $k + 1$ non-isolated vertices. Otherwise the answer is trivial.

For such a graph G we construct a bipartite graph G' as follows; see Figure 2.2. For every vertex $v \in V$ we construct a vertex s_v of V' and for every edge $e = (u, v) \in E$ we construct a vertex s_e of V'. For edge $e = (u, v)$ we construct two edges (s_u, s_e) and (s_v, s_e) in E'.

Additionally, we construct $i = 1, 2, \ldots, k - 1$ columns of k vertices where each vertex of column i has an edges to every vertex on column $i + 1$. For every vertex of column $k - 1$ there is an edge to every $s_v \in V'$. Finally, there is a unique vertex r that is connected to all vertices of the first column.

The construction means that after $k - 1$ steps from r at least one vertex of column $k - 1$ is burning and threatens all vertices s_v.

So if we protect a k-Clique of V by choosing the corresponding vertices $s_v \in V'$ in the first k steps, we will protect the k-Clique vertices and also the $\binom{k}{2}$ (edge) vertices s_e of V' of the k-Clique. In an additional step $k + 1$ we can save one more edge vertex s_e. This vertex exists because we assumed that $k + 1$ vertices in G are non-isolated. Thus, if a k-Clique exists, we can save $k' = k + \binom{k}{2} + 1$ vertices.

So we would like to answer the decision problem for G', root r and k'. The graph G' is constructed in polynomial time.

Conversely, suppose there is a strategy that saves at least k' vertices. After k time steps the fire will always reach the vertices s_v since the $k - 1$ columns contain k vertices. For the first $t \geq k - 1$ steps, it is not helpful to protect a vertex in one of the columns from 1 to $k - 1$ because they only will protect themself and we can also choose one of s_v or s_e instead. This also means that after k steps, all vertices except those chosen by the strategy are burning. Protecting one of the vertices s_e before step k is also needless because it only saves itself. Therefore the best one can do is, choose k vertices s_v in the first k steps. These k vertices can save at most $\binom{k}{2}$ vertices, which is only possible if the chosen vertices build a k-Clique in G. □

2.2 Trees

2.2.1 Greedy approximation for a tree

A greedy algorithm for a tree always protects the subtree with the largest number of vertices. Figure 2.3 shows that this strategy is not better than a $1/2$ approximation. The optimal strategy protects $2(k - 1)$ vertices whereas greedy protects only $k + 1$ vertices. Note that the greedy algorithm always choose a vertex neighboring a contaminated vertex but also the optimal does as we will prove later.

Theorem 11 *For a problem instance $(T, r, 1)$ of a rooted tree $T = (V, E)$ the greedy strategy gives a $\frac{1}{2}$ approximation for the optimal number of vertices protected. This bound is tight.*

Proof. As shown by Figure 2.3 greedy is not better than $\frac{k+1}{2(k-1)} \mapsto \frac{1}{2}$.

For the upper bound we can subdivide the greedy strategy S into steps that outperform the current optimal move and steps that do not outperform the current optimal move. Outperforming
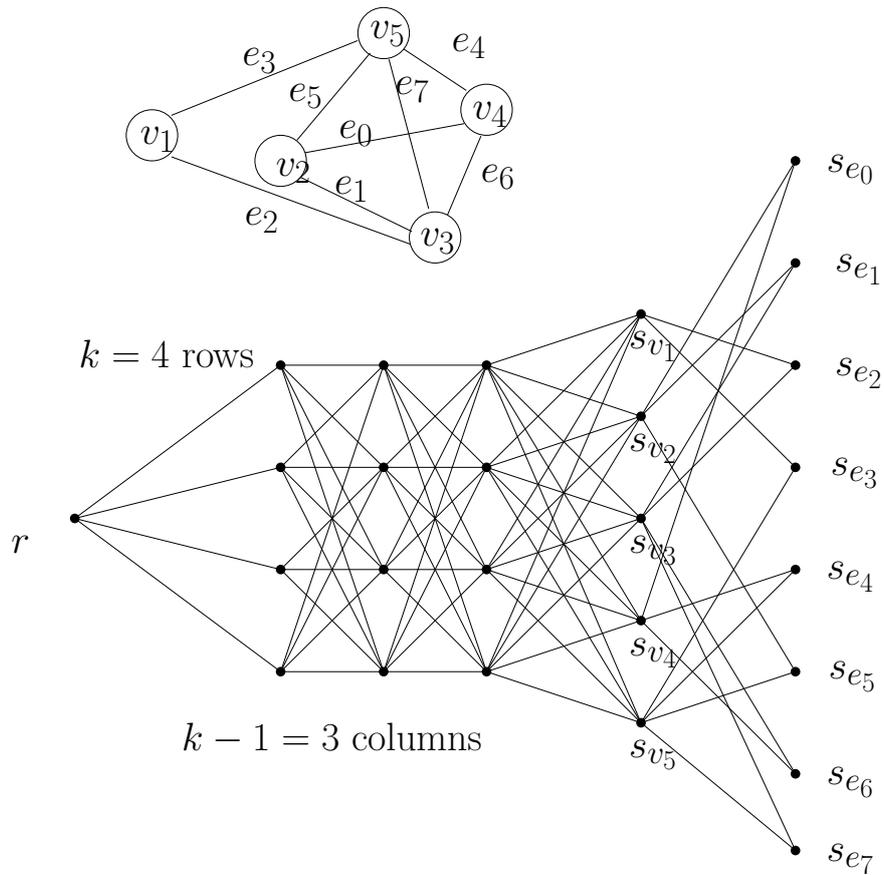
Figure 2.2: Protecting the vertices of the k-Clique in k steps and an additional vertex s_{e_j} gives $k' = k + \binom{k}{2} + 1$ protected vertices in total.
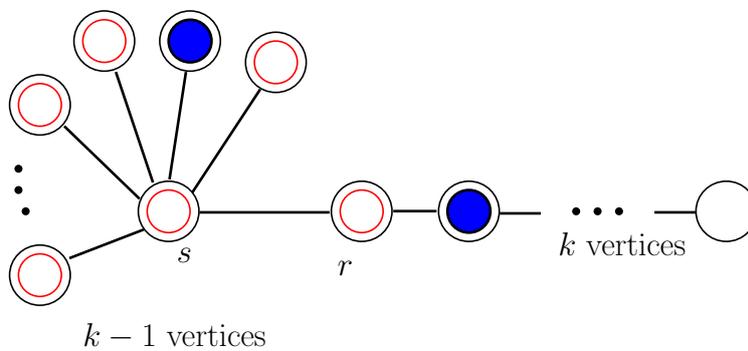

Figure 2.3: The greedy algorithm does not protect the vertex s first and only saves $k+1$ vertices.

means saving at least as many vertices as the optimal move in the same step. For an optimal strategy opt let opt_A (greedy is not worse) denote the first category of safe vertices and let opt_B (greedy is worse) denote the second category of safe vertices. Let S_G denote the number of vertices saved by S. We can assume that each step of a strategy cuts of a subtree of T and as long as the fire spreads the distance of burning vertices to the root increases by 1 in every time step.

We would like to show that $2S_G \geq \text{opt}_A + \text{opt}_B$ holds. This can be seen as follows. Greedy starts with a step as least as good as the optimal one because it is greedy. Let us assume that at some moment in time the optimal move is better than a greedy move or there is no greedy move anymore. Then in one of the previous steps greedy has chosen a predecessor of the optimal move v, otherwise the greedy algorithm could have also picked the optimal move now because of the distance of vertex v to the root. Thus the vertices saved by the optimal move now have already been saved by greedy before. $\square$

Exercise 6 *Give an example for the proof idea of Theorem 11. Show that Theorem 11 also holds for the case (T, r, p) for $p > 1$.*

2.2.2 Exponential time algorithm for general trees

The decision problem is already NP-complete for trees. In this subsection we show that the problem is fixed-parameter-tractable by giving a somewhat efficient algorithm.

For this purpose we also introduce another variant of the game. How many vertices can be saved, if only k guards can be placed.

Firefighter Decision Problem (Protection by k guards):
Instance: A Graph $G = (V, E)$ of degree d with root vertex r and p firefigther per step and an integer k.
Question: What is the strategy that saves a maximum number of vertices by protecting k vertices in total?

The algorithm is based on the following structural property for a strategy on trees.

Lemma 12 *For any optimal strategy for an instance of the firefigther decision problem on trees (protection by k guards, saving k vertices) the vertex defended at each time is adjacent to a burning vertex. There is an integer l, so that all protected vertices have depth at most l, exactly one vertex p_i at each depth is protected and all ancestors of p_i are burning.*

Proof. If the vertex defended at some time step t has no burning neighbor, it is possible to improve the strategy by protecting the vertex closer to the root. Thus, in any time step t one vertex at depth t is chosen. $\square$

Exercise 7 *Show an example that Lemma 12 does not hold for a a general graph.*

Now we present an efficient algorithm that computes an optimal strategy by dynamic programming for the above maximum protection-by-k-guards setting. An example is given in Figure 2.4.

Starting from the root vertex we label the vertices in pre-order. By the pre-order we define the relation v *lies to the left of* w by $\text{pre}(v) < pre(w)$. Due to Lemma 12 it suffices to consider the vertices up to depth k only.

For the dynamic program process we more generally consider a substrategy where we place a guard at step t or we do not place a guard at step t. This is due to the fact, that in step t we cut of a subtree, consider a smaller tree and assume that in this remaining tree no guard was placed
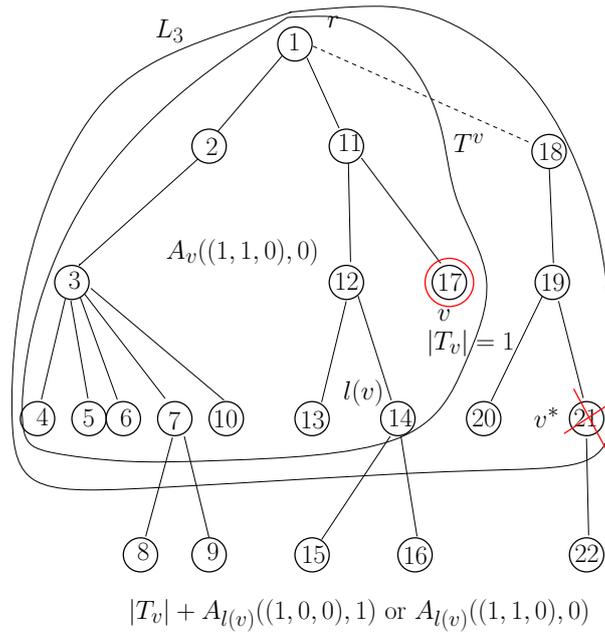
Figure 2.4: The pre-order of a tree T, the set L_k of vertices of depth $\leq k$. Computing $A_v((1,1,0),0)$ means that we are searching for a strategy in the tree T^v that sets guards in the first and second depth and if a vertex is protected along the path from r to v, then its depth is greater than 0. For the recursion we consider two cases. If the vertex v will be protected, we are looking for $|T_v| + A_{l(v)}((1,0,0),1)$. Here the second parameter 1 says that we are not allowed to block a predecessor of v any more in this case. If the vertex v will be not protected, we are looking for $A_{l(v)}((1,1,0),0)$. The maximum of both is the best choice.

at step t. Therefore, we have a vector $X \in \{0,1\}^k$ that represents a general strategy. $X(j) = 1$ denotes the case that at step j a guard will be placed at depth j and $X(j) = 0$ denotes the case where no guard is placed at step j.

Let L_k be the set of vertices of the tree T with depth $\leq k$. For any $v \in L_k$, we consider the subtree T^v of T with vertices from L_k and pre-order at most $\mathrm{pre}(v)$ (all vertices of L_k to the left of v including v). We would like to compute the maximum number $A_v(X)$ of vertices that can be saved in T_v if we apply a strategy for up to k steps that behaves as indicated by the vector X.

There will be a rightmost vertex v^* of depth k in T and we are obviously interested in $A_{v^*}(1^k)$. This is the end value we would like to compute.

Let us try to explain the recursive formula for a vertex v in general. By dynamic programming in step j we can either choose vertex v of depth j to be protected in step j or not. Let $|T_v|$ denote the size of the subtree of T starting with root v.

If v is protected, we have saved $|T_v|$ vertices cut of the tree T_v, exchange the entry j of the current vector X by 0 and turn over to the next entry in L_k with pre-order less than v. For vector X and vertex v of depth j let, X^v denote the $0-1$ vector where the entry of X at j is set to 0. Let $l(v)$ denote the vertex of L_k with largest pre-order smaller than v. So in the recursion just consider to compute $A_{l(v)}(X^v)$ and add $|T_v|$ in this case.

There is a problem because in the subtree $T^{l(v)}$ it might be allowed to place a guard on the shortest path from r to v which is no more allowed, if v was protected before. Therefore we introduce a second variable i and always compute $A_v(X,i)$ where i denotes the depth along the path from r to v after that a vertex can be chosen for protection. In the above case we have to

compute $A_{l(v)}(X^v, \text{depth}(v) - 1)$.

We can assume that at v the recursion was started with $A_v(X, i)$. And the above case only have to be taken into account if $X(\text{depth}(v)) = 1$ and $\text{depth}(v) > i$.

Now let us assume that v will not be protected. In this case for $A_v(X, i)$ we do not change X and turn over to $A_{l(v)}$ for the predecessor $l(v)$ of v in L_k. The parent of v of depth $(\text{depth}(v) - 1)$ is always on the path of r to $l(v)$. If it was only allowed to set a guard at depth larger than i on the path from r to v, we only have to take care that i is not larger than $\text{depth}(v) - 1$. This means that $A_{l(v)}(X, \min(\text{depth}(v) - 1, i))$ has to be computed.

Theorem 13 *Computing the optimal protection strategy for k guards on a tree T of size n can be done in $O(n2^k k)$ time.*

Proof. By the above definitions we consider a dynamic programming approach with

$$A_v(X, i) = \max \left\{ \begin{array}{l} A_{l(v)}(X, \min(\text{depth}(v) - 1, i)) \\[2mm] [X(\text{depth}(v)) = 1 \text{ and } \text{depth}(v) > i] \cdot \left(|T_v| + A_{l(v)}(X^v, \text{depth}(v) - 1) \right) \end{array} \right\}$$

where $[\phi]$ equals 1 if ϕ holds true and 0 otherwise.

For $v \in L_k$ the value $A_v(X, i)$ just denotes the optimal protection number for a strategy in T^v that sets a number of guards on each depth w.r.t. the entries in X and does not set a guard on the path from r to v before or at depth i.

We are simply searching for $A_{v^*}(1^k, 0)$ where vertex v^* is the rightmost vertex of depth k in T.

By the above description the dynamic programming procedure is correct. We compute L_k, $l(v)$ and $|T_v|$ for each vertex v in linear time by a pre-order walkthrough. Then we traverse the vertices of L_k from left to right and we have at most $n \times 2^k \times k$ entries $A_v(X, i)$ where n stands for v, 2^k stands for X and k stands for i. $\square$

Exercise 8 *Compute the best strategy of the example in Figure 2.4 for $k = 2$ and with the dynamic programming approach introduced above.*

Exercise 9 *Consider a binary tree and design an optimal strategy. Analyse the time complexity for computing the strategy.*

Exercise 10 *For a given tree, formalize the computation of an optimal strategy by an Integer Linear Program.*

With the above algorithm we can also answer the question whether k vertices can be saved.

Corollary 14 *Computing a strategy for a tree T of size n that saves at least k vertices can be done in $O(n2^k k)$ time.*

Proof. If (T, r, k) is an instance for the saving-k-vertices variant. It is clear that a strategy with k placements will solve the problem. Therefore it suffices to run the above algorithm for $i = 1, \ldots, k$. We can save k vertices if and only if we can save at least k vertices for some value of i.

Thus we have

$$\sum_{i=1}^{k} i 2^i n \leq kn \sum_{i=1}^{k} 2^i = (2^{k+1} - 2)kn$$

so that the worst-case running time is in $O(n2^k k)$.

<div align="right">□</div>

Finally, we would like to build up a good subexponential time bound and make use of the following structural property in order to find a good bound on k w.r.t. n.

Lemma 15 *If a vertex at depth d is burning in an optimal strategy for an instance of the firefigther problem on trees, at least $\frac{1}{2}(d^2 + d)$ vertices are safe.*

Proof. Let us assume that in T and for an optimal startegy, a vertex v at depth d is burning. Then by Lemma 12 there is a protected vertex v_i for any depth $i = 1, \ldots, d$. Any tree T_{v_i} should contain at least $d - i + 1$ vertices. Otherwise it was better to choose a vertex along the path from r to v in this step i. Thus

$$\sum_{i=1}^{d}(d - i + 1) = \frac{1}{2}(d^2 + d)$$

gives the bound.

<div align="right">□</div>

Theorem 16 *There is an $O\left(2^{\sqrt{2n}}n^{3/2}\right)$ algorithm for the firefigther problem on a tree of size n.*

Proof. We show that we can run the algorithm of Theorem 13 for $k \leq \sqrt{2n}$. Suppose a vertex of depth $\sqrt{2n}$ is burning. Then by Lemma 15 $n + \sqrt{n/2} > n$ vertices are safe witch contradicts the number n of vertices. This means that all vertices of depth $\sqrt{2n}$ are safe in an optimal strategy. In turn an optimal strategy makes use of less than $\sqrt{2n}$ guards. Thus we set $k \leq \sqrt{2n}$ which gives the bound.

<div align="right">□</div>

2.2.3   Capture of an Intruder by moving agents

Up to now we have considered stationary guards that check and block vertices of the graph or tree. Many other variants are known from the literature. We discuss the following version. A set of k agents starts at a homebase vertex b in a given tree $T = (V, E)$ and there is some intruder somewhere in T. The following problem definition and its solution goes back to Barrière et al. [].

The intention is to *clear* all *edges* of the tree. A *contiguous search strategy* can perform one of the following two operations in one search time step:

1. Place a team of p guards on a vertex.

2. Move a team of m guards along an edge.

Additionally, the set of all *cleared* edges E_i after step i has to be connected for any i.

In one time step a set of agents located at vertex u can move along an edge $e = (v, u)$ from v to u. There is an integer edge weight $w(e) \geq 1$ that says how many agents have to move along e so that the intruder cannot cross the edge from u to v in this step. In this sense the agents *clear* the link in this case. Intuitively, we can think of a large corridor and cleaning the corridor from v to u requires at least $w(e)$ agents. Furthermore, we have integer weights $w(v)$ for the vertices. Locally, a clear link $e = (v, u)$ is preserved from recontamination, if either the vertices v and u are guarded by $w(v)$ and $w(u)$ agents, respectively or all other edges incident to e are also clear.
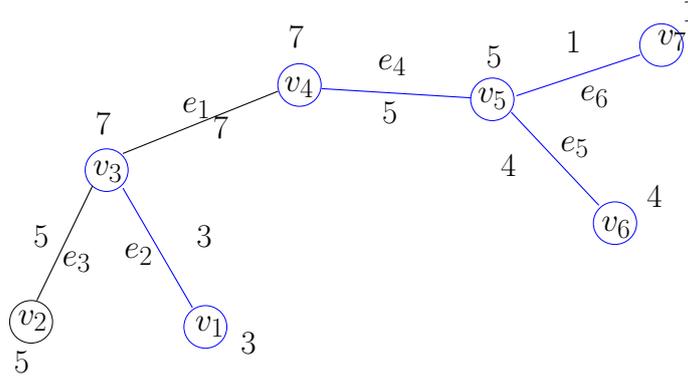
Figure 2.5: A given weighted tree $T = (V, E)$. A successful contiguous strategy can start with 10 agents at v_1, also $\text{cs}(T) = 10$ holds. The size of the frontiers of $X_1 = \{e_4, e_5, e_6\}$ and $X_2 = \{e_2\}$ is $w(X_1) = 7$ and $w(X_2) = 10$, respectively.

We assume that guarding a vertex is at least as hard as the maximum number of searchers required for clearing an incident edge. Thus, we require $w(v) \geq w(e)$ for any $e = (v, u) \in E$. So for any vertex $v \in V$ the weight could for example be given by $w(v) := \max_{e=(u,v)\in E} w(e)$. An example for a weighted tree is given in Figure 2.5.

In any time step (after a step of the search strategy) an edge e becomes recontaminated, if there is a path from a contaminated edge e' to edge e that is not blocked by agents on the vertices. This means that a corresponding intruder has infinite speed for crossing an arbitrary number of edges. We consider contiguous search strategies as described above and ask for the minimum number of guards, $\text{cs}(T)$, that suffices to finally clear all links of the tree T.

A contiguous strategy for Figure 2.5 can start with 10 agents in vertex v_7, clears edge e_6. Leaves 6 agents at v_5 and clear e_5 with 4 agents. Moves back with 4 agents to v_5 and clears edge e_4 and edge e_7 with 10 agents. Now, we leave 7 agents at v_3, clear e_2 by 3 agents, move back with three agents and finally clear the last edge e_3. Note that in the second last step, if we leave less than 7 agents at v_3, the edge e_3 is contaminated and can recontaminate the full tree.

We would like to consider such monotone strategies. That is, a link that has already been cleaned at some step t should not be contaminated (or visited by the intruder) later again. This means that some of the agents will become stationary for a period of time in order to block the movement of the intruder (or the recontamination of links). Additionally, we can compute optimal monotone strategies efficiently as we will see below. First, we show that monotone strategies always exist.

2.2.4 Existance of monotone strategies

One main structural result is, that for a tree T for the computation of $\text{cs}(T)$ it suffices to consider a montone strategy, where all agents start at the same homebase vertex b.

Theorem 17 *For any weighted tree T there is a monotone contiguous search strategy with $\text{cs}(T)$ agents where all agents initially start at the same vertex b.*

For the proof we require some notations. Let $T = (V, E)$. For a subset $X \subseteq E$ we denote all vertices that have a vertex incident to X and $E \setminus X$ as the *boundary vertices* $\delta(X)$. If X_i denotes the set of clear links after time step i of a strategy then at least

$$w(X_i) := \sum_{v \in \delta(X_i)} w(v) \tag{2.1}$$
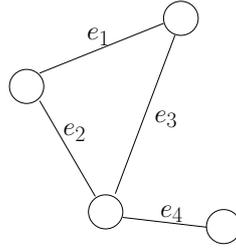
Figure 2.6: Crusades are only defined by subsets of edges of a graph. For the given graph G there is a connected crusade $(\emptyset, \{e_1\}, \{e_1, e_2\}, \{e_2\}, \{e_2, e_3\}, \{e_1, e_2, e_3\}, \{e_3, e_4\}, \{e_1, e_3, e_4\}, \{e_1, e_2, e_3, e_4\})$ and a pogressive connected crusade is $(\emptyset, \{e_1\}, \{e_1, e_2\}, \{e_1, e_2, e_3\}, \{e_1, e_2, e_3, e_4\})$.

guards have been used. (Note that the contamination threatens not only the directly adjacent links but also any fully non-protected path!)

In Figure 2.5 we have $w(\{e_4, e_5, e_6\}) = 7$ and $w(\{e_2\}) = 10$.

For $G = (V, E)$ a sequence $(X_0, X_1, \ldots, X_m)$ of subsets $X_i \subseteq E$ is called a *crusade*, if $X_0 = \emptyset$ and $X_m = E$ and $|X_i \setminus X_{i-1}| \leq 1$ for $1 \leq i \leq m$. The *frontier* of a crusade $(X_0, X_1, \ldots, X_m)$ is given by $\max_{1 \leq i \leq m} w(X_i)$ as defined in Equation 2.1 above, which is the minimum number of guards required for defending any X_i. A crusade is *progressive* if $X_0 \subseteq X_1 \subseteq \cdots \subseteq X_m$ and $|X_i \setminus X_{i-1}| = 1$ for $1 \leq i \leq m$. The crusade is *connected*, if X_i is connected for $1 \leq i \leq m$.

Note that the definition of crusades holds for arbitrary graphs G. See Figure 2.6 for an example.

There will be a progressive connected crusade that will finally describe a strategy. In the very beginning it is allowed to set agents on the vertices in many non-crusading steps. Then the crusade starts to clean the links. But keep in mind that the above (connected) crusades are only defined over sets of links up to now and do not directly represent a strategy.

Exercise 11 *Define a crusade $C = (X_0, X_1, \ldots, X_m)$ of a tree T that is not related to a strategy.*

It is easy to see, that for $\mathrm{cs}(T) \leq k$ there should be a connected crusade of frontier $\leq k$ in T. For a given contiguous strategy S let $C = (X_0, X_1, \ldots, X_m)$ denote the sequence of clear links after each search step i. In any search step we can clear at most one additional edge, which means $|X_i \setminus X_{i-1}| \leq 1$ for $1 \leq i \leq m$. Since X_i is connected (definition of contiguous) and is no further destructed after search step i, we have $w(X_i) \leq k$. Of course by construction we have $X_0 = \emptyset$ and $X_m = E$. An example for Figure 2.5 is the above mentioned strategy and the connected crusade $C = (\emptyset, \{e_6\}, \{e_6, e_5\}, \{e_6, e_5, e_4\}, \{e_6, e_5, e_4, e_1\}, \{e_6, e_5, e_4, e_1, e_2\}, \{e_6, e_5, e_4, e_1, e_2, e_3\})$

We conclude:

Lemma 18 *For $\mathrm{cs}(T) \leq k$ there is a connected crusade of frontier at most k.*

Now we would like to show that there is also a progressive connected crusade in T of frontier at most k. Note that the above connected crusade of Figure 2.5 is indeed progressive.

So starting from the above Lemma from all connected crusades $C = (X_0, X_1, \ldots, X_m)$ of frontier at most k we choose one that satisfies the following properties:

1. $\sum_{i=0}^{m} (w(X_i) + 1)$ is minimum.

2. Amog all crusade satisfying condition 1. choose one with: $\sum_{i=0}^{m} |X_i|$ is minimum.

Obviously, one such crusade has to exist. So we only have to show that the crusade is progressive. This means that we have to show $X_0 \subseteq X_1 \subseteq \cdots \subseteq X_m$ and $|X_i \setminus X_{i-1}| = 1$ for $1 \leq i \leq m$.

Let us first assume that $|X_i \setminus X_{i-1}| = 0$ holds, which means that $X_i \subseteq X_{i-1}$. Therefore, we consider the connected crusade

$$C' = (X_0, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m)$$

of frontier at most k which contradicts condition 1. Note that $|X_{i+1} \setminus X_{i-1}| \leq 1$ can be concluded from $|X_{i+1} \setminus X_i| \leq 1$ and $X_i \subseteq X_{i-1}$.

Thus, we can assume that $|X_i \setminus X_{i-1}| = 1$ holds for $1 \leq i \leq m$. Next, we prove $X_{i-1} \subseteq X_i$ for $1 \leq i \leq m$. We will first conclude that

$$w(X_{i-1} \cup X_i) \geq w(X_i) \tag{2.2}$$

holds for $1 \leq i \leq m$. Otherwise, we make use of the crusade

$$C' = (X_0, \ldots, X_{i-1}, X_{i-1} \cup X_i, X_{i+1}, \ldots, X_m) \tag{2.3}$$

of frontier at most k which is a contradiction to condition 1. We know that X_i and X_{i-1} are connected. Thus, the set $X_{i-1} \cup X_i$ is connected since $|X_i \setminus X_{i-1}| = 1$ holds. We can also conclude that $|X_{i+1} \setminus (X_{i-1} \cup X_i)| \leq 1$ holds, since $|X_{i+1} \setminus X_i| = 1$. If $|X_{i+1} \setminus (X_{i-1} \cup X_i)| = 0$ holds we can go back to the first step again.

Now assume that Equation 2.2 holds. For any two arbitrary link sets A and B of T we have $w(A \cup B) + w(A \cap B) \leq w(A) + w(B)$ which is the question of Exercise 12. We conclude $w(X_{i-1} \cap X_i) \leq w(X_i)$ for $1 \leq i \leq m$ from Equation 2.2 and consider

$$C'' = (X_0, \ldots, X_{i-2}, X_{i-1} \cap X_i, X_{i+1}, \ldots, X_m). \tag{2.4}$$

Now, C'' has frontier at most k.

By the minimality of C w.r.t. condition 2. we conclude that $|X_{i-1} \cap X_i| \geq |X_{i-1}|$ and therefore $X_{i-1} \subseteq X_i$ holds. We conclude $|X_i \setminus (X_i \cap X_{i-1})| = |X_i \setminus X_{i-1}| = 1$ and $|(X_i \cap X_{i-1}) \setminus X_{i-2}| \leq |X_{i-1} \setminus X_{i-2}| \leq 1$.

It remains to show that $X_{i-1} \cap X_i$ is also connected. Assume that this is not the case. Let $\{e\} = X_i \setminus X_{i-1}$ and $W = X_{i-1} \setminus X_i$ and $Z = X_{i-1} \cap X_i$. By assumption $Z = Z' \cup Z''$ where Z' and Z'' do not share a vertex. The situation is depicted in Figure 2.7. We have $X_{i-1} = Z' \cup Z'' \cup W$ and $X_i = Z' \cup Z'' \cup \{e\}$. Since X_i and X_{i-1} are both connected, and $\{e\} \notin W$ there is a cycle $Z' \cup Z'' \cup W \cup \{e\}$ in T which contradicts to the assumption that T is a tree, $X_{i-1} \cap X_i$ is also connected. We have proven the following Lemma.

Lemma 19 *For $cs(T) \leq k$ there is a progressive connected crusade of frontier at most k in T.*

Exercise 12 *For two link (edge) sets A and B in a graph G prove that $w(A \cup B) + w(A \cap B) \leq w(A) + w(B)$ holds.*

Now, for obtaining a monotone contiguous strategy from above progressive connected crusade, we first extend the tree T. We replace every link e by two consecutive links e' and e'' of the same weight $w(e)$. The newly inserted vertex v has also weight $w(v) := w(e)$. This means that after this transformation any link has at least one vertex of degree 2. Let T' be the outcome of this process. We will use this property for designing a monotone contiguous strategy from a progressive connected crusade. Afterwards we can transform the strategy back for acting on T. Obviously, also $cs(T') \leq cs(T)$ holds.
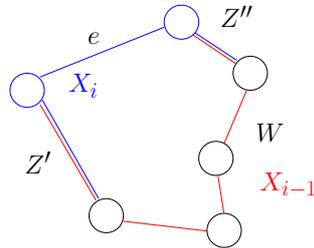
Figure 2.7: Let $W = X_{i-1} \setminus X_i$ and $\{e\} = X_i \setminus X_{i-1}$ and $Z = X_{i-1} \cap X_i$. By assuption $Z = Z' \cup Z''$ where Z' and Z'' do not share a vertex. Thus, the cycle $Z' \cup Z'' \cup W \cup \{e\}$ contradicts T.
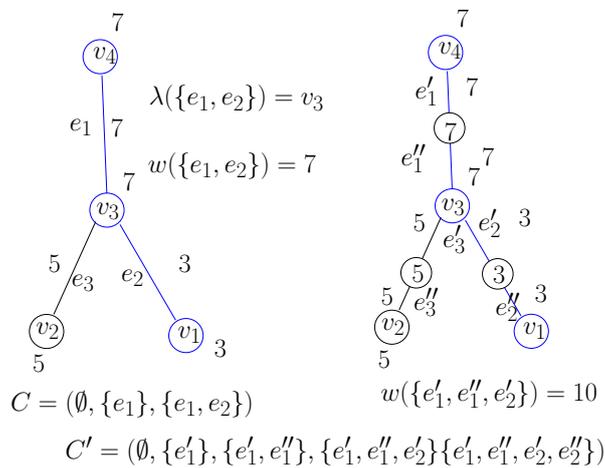
Figure 2.8: Replacing any link e in the tree T by two consecutive links e' and e'' yields the desired correspondance of the frontier and the frontier and the number of agents required for $C = (\emptyset, \{e_1\}, \{e_1, e_2\})$.
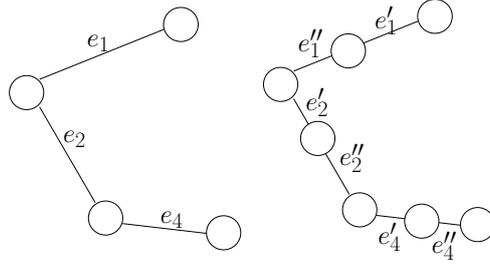
Figure 2.9: Replacing any link e in the tree T by two consecutive links e' and e''. Any link in T' has at least one vertex of degree 2.

The main reason for this enlargement is presented in Figure 2.8. Unfortunately, there is no one to one correspondance between the number of agents required for a contiguous strategy and the size of the frontier of connected progressive crusade. For example, in Figure 2.8 we require 10 agents for successively cleaning $C = (\emptyset, \{e_1\}, \{e_1, e_2\})$ but the frontier of C is 7. If we enlarge the tree by artificial edges, we get the required correspondance for $C' = (\emptyset, \{e'_1\}, \{e'_1, e''_1\}, \{e'_1, e''_1, e'_2\}\{e'_1, e''_1, e'_2, e''_2\})$. Here $w(\{e'_1, e''_1, e'_2\}) = 10$ gives the frontier.

Lemma 20 *Let T' be a tree so that every link has at least one vertex of degree 2. If there is a progressive connected crusade of frontier $\leq k$ in T', there is a monotone contiguous search strategy using $\leq k$ guards in T' and the guards can be initially placed at a single vertex v_1.*

Proof. For the progressive connected crusade $C = (X_0, X_1, \ldots, X_m)$ of frontier $\leq k$ and $e_i = (v_i, u_i) := X_i \setminus X_{i-1}$ we construct a strategy that clears the links $e_1, \ldots, e_m$ successively.

In the beginning we set k guards at v_1. We have $w(X_1) = w(v_1) + w(u_1) \leq k$ and also $w(e_1) \leq w(u_1)$ and move $w(u_1)$ searchers along w_1.

Now by induction let us assume that we have constructed a monotone contiguous strategy for $i-1$ links $e_1, \ldots, e_{i-1}$ without recontaminations. Consider the edge $e_i = (v_i, u_i)$ incident to X_{i-1}, say $v_i \in \delta(X_{i-1})$.

If $w(X_{i-1}) + w(u_i) \leq k$ holds, $w(u_i)$ agents can move from v_i along e_i to u_i and clears link e_i and we are done.

Therefore, assume that $w(X_{i-1}) + w(u_i) > k$ holds. In this case not both vertices of e_i can be in $\delta(X_i)$. We have $v_i \in \delta(X_{i-1})$. Assume $v_i \in \delta(X_i)$. We conclude $\deg(v_i) > 2$ and $\deg(u_i) = 2$. This means that $u_i \in \delta(X_i)$ implies that the other link $f_i \neq e_i$ that contains u_i has to be contaminated, and $u_i \notin \delta(X_{i-1})$. Therefore, $w(X_i) = w(X_{i-1}) + w(u_i)$, but this has to be $\leq k$, a contradiction.

For $w(X_{i-1}) + w(u_i) > k$ at most one vertex of (v_i, u_i) is in $\delta(X_i)$ and we consider the corresponding cases; see also Figure 2.10.

1. $v_i \in \delta(X_i), u_i \notin \delta(X_i)$: As shown above $\deg(u_i) = 2$ and the other link $f_i \neq e_i$ that contains u_i belongs already to X_{i-1}. There are already $w(u_i)$ guards at u_i in step $i-1$ and they can clear the edge from u_i to v_i.

2. $v_i \notin \delta(X_i), u_i \notin \delta(X_i)$: This means that e_i is the only contaminated edge adjacent to v_i and u_i. We can move with $w(v_i)$ searchers from v_i to u_i.

3. $v_i \notin \delta(X_i), u_i \in \delta(X_i)$: This means that $w(X_i) = w(X_{i-1}) - w(v_i) + w(u_i)$ and we have at least $w(v_i)$ guards at v_i. Move all $k - w(X_{i-1})$ free guards to v_i. We have $w(v_i) + k - w(X_{i-1}) \geq w(v_i) + w(X_i) - w(X_{i-1}) \geq w(u_i)$ agents at v_i that can clear e_i now.
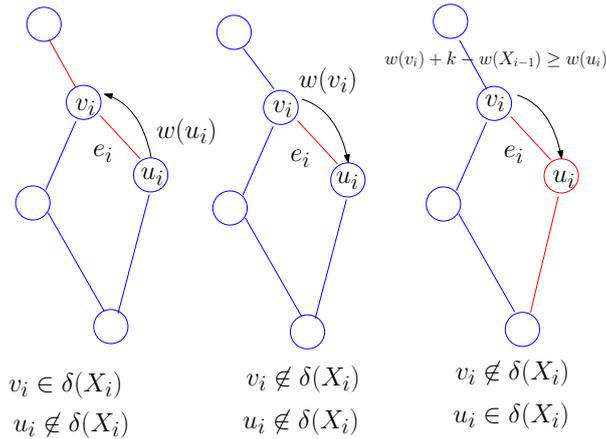
Figure 2.10: Not both vertices v_i and u_i can be in $\delta(X_i)$. Therefore, also edge e_i can be cleaned.

Altogether, there is a successful montone contiguous strategy for T' with k agents adapted from the progressive connected crusade $C = (X_0, X_1, \ldots, X_m)$ of frontier $\leq k$. $\square$

Note, that the above Lemma also holds for graphs with the same properties. The same holds for the following Lemma where we show that we can obtain a strategy for T from the strategy for T'.

Lemma 21 *Any contiguous monotone strategy for T' can be translated to a contiguous monotone strategy for T with the same number k of agents.*

Proof. Let $e' = (x, y)$ and $e'' = (y, z)$ be links stemming from the extension of a link e. If q guards move from x to y or z to y, they stay in there place in T. If q guards move from y to x or from y to z, they will move from z to x or from x to z in T, respectively. $\square$

The other way round, any strategy for T is also a strategy for T'.

Lemma 22 *Any contiguous monotone strategy for T with k agents can be translated to a contiguous monotone strategy for T' with the same number k of agents.*

Proof. A move along an edge e in T is splitted into two moves along e' and e'' in T'. If the move clears e, then $q \geq w(e)$ have traversed e. From the construction q searchers are also enough for $w(e) = w(e') = w(e'')$ and the weight $w(e)$ of the intermediate vertex. $\square$

We collect our results:

Proof of Theorem 17: From Lemma 21 we conclude $\text{cs}(T') \leq \text{cs}(T)$. From Lemma 18 we obtain a connected crusade of frontier $\leq \text{cs}(T)$ in T'. From Lemma 19 we conclude that there is a progressive connected crusade of frontier $\leq \text{cs}(T)$ in T'. From Lemma 20 we obtain a monotone contiguous search strategy using $\leq \text{cs}(T)$ guards in T' and we can assume that all searchers are initially at a single starting vertex v_1. From Lemma 22 we conclude that there is also an optimal monotone contiguous search strategy that starts with all guards in a single vertex.

2.2.5   Designing a monotone strategy for unit weights

By Theorem 17 we can start a strategy from a single vertex v and we can consider monotone strategies. Therefore, we design an optimal strategy for any starting vertex v and for the rooted tree T_v we compute the minimum number, $\text{cs}(T_v)$, of agents required for starting in v. Finally we have $\text{cs}(T) = \min_{v \in T} \text{cs}(T_v)$.
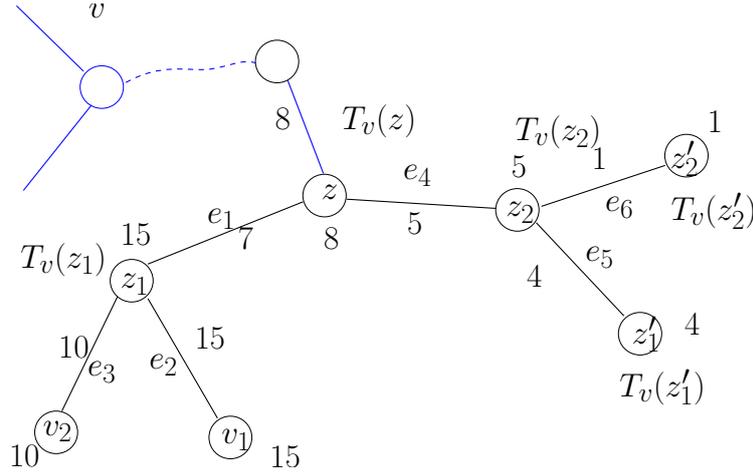
Figure 2.11: The rooted tree T_v has two subtrees $T_v(z_1)$ and $T_v(z_2)$ at vertex z. We have $cs(T_v(z_1)) = 25$ and $cs(T_v(z_2) = 6$ and $cs(T_v(z_2)) + w(z) = 14 < 25 = cs(T_v(z_1))$ which means that $cs(T_v(z)) = 25$ holds. We leave $w(z)$ agents at z and clean $T_v(z_2)$ first. In $T_v(z_2)$ the same situation occurs, here $cs(T_v(z_1')) = 4$ and $cs(T_v(z_2)) = 1$ but $cs(T_v(z_2')) + w(z_2) = 6 > 4 = cs(T_v(z_1')$. Therefore we require $cs(T_v(z_2')) + w(z_2) = 6$ agents, first we clean $T_v(z_2')$ by 1 agent and block z_2 by 5 agents. Then we clean $T_v(z_1')$ by 6 agents.

An optimal monotone strategy for computing, $cs(T_v)$, will also give an ordering of all vertices z of T_v, stating which subtree, say $T_v(z)$, of T_v w.r.t. root v is fully cleared first. We can also consider the subtree $T_v(z)$ with root z by its own and ask for $cs(T_v(z))$ for short and an optimal monotone strategy.

We denote the children of the vertex z of the subtree $T_v(z)$ of T_v by $z_1, \ldots, z_d$ w.r.t. the order $cs(T_v(z_i)) \geq cs(T_v(z_{i+1}))$ for $i = 1, \ldots, d-1$. An example is given in Figure 2.11. Now, we can prove the main structural result. Unfortunately, there is a flaw in the proof of Barrière at al. and we can only prove the statement for unit weighted trees. The flaw is precisely marked in the proof below and also shown in Figure 2.12.

Lemma 23 *Let $z_1, \ldots, z_d$ be the $d \geq 2$ children of a vertex z in T_v and assume that $cs(T_v(z_i)) \geq cs(T_v(z_{i+1}))$ for $i = 1, \ldots, d-1$. We have*

$$cs(T_v(z)) = \max\{cs(T_v(z_1)), cs(T_v(z_2)) + w(z)\} \qquad (2.5)$$

if the tree T is a tree with unit weights.

Proof. We can assume that $cs(T_v(z)) \geq cs(T_v(z_1))$ holds because we have to clear $T_v(z_1)$ before clearing $T_v(z)$. If in Equation 2.5 $cs(T_v(z_1)) \geq cs(T_v(z_2)) + w(z)$ holds, we can clear $T_v(z)$ by setting $w(z)$ on z and clear all $T_v(z_i)$ by $cs(T_v(z_1))$ agents but $T_v(z_1)$ last. Altogether, $cs(T_v(z_1))$ agents are required and they are sufficient.

So let us assume that in Equation 2.5 $cs(T_v(z_1)) < cs(T_v(z_2)) + w(z)$ holds. We would like to prove that $cs(T_v(z_2)) + w(z) - 1$ agents are not sufficient. We consider two cases:

1. $T_v(z_2)$ **is cleared before** $T_v(z_1)$**:** While $cs(T_v(z_2))$ agents clear $T_v(z_2)$ there are only $w(z) - 1 = 0$ agents left for blocking a vertex in $T_v(z_1)$. Recontamination!

2. $T_v(z_1)$ **is cleared before** $T_v(z_2)$**):** While $cs(T_v(z_1))$ agents clear $T_v(z_1)$ there are no more than $w(z) - 1 = 0$ agents left for blocking a vertex in $T_v(z_2)$ (because $cs(T_v(z_1)) = cs(T_v(z_2))$). Recontamination!
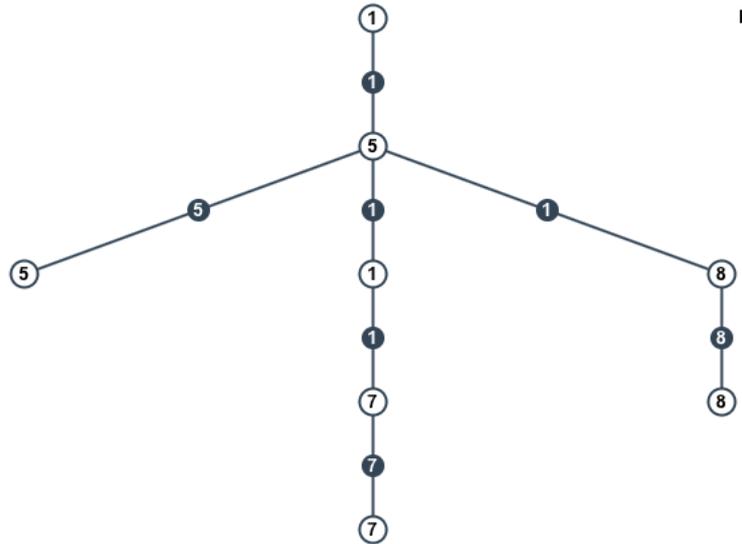
Figure 2.12: The flaw in the prove of Barriére et al. The recursion $\mathrm{cs}(T_v(z)) = \max\{\mathrm{cs}(T_v(z_1)), \mathrm{cs}(T_v(z_2)) + w(z)\}$ does not hold for arbitrary weighted trees.

General weighted trees: The above statement does not hold for general weighted trees, because the fact that an agent could only partially decontaminates $T_v(z_2)$ or $T_v(z_1)$ is not taken into account. For example, consider the vertex, say v with weight 5 in the center of Figure 2.12. and let z_1, z_2, and z_3 be the children of v below v from right to left. We have $\max\{\mathrm{cs}(T_v(z_1)), \mathrm{cs}(T_v(z_2)) + w(z)\} = \max\{8, 7+5\} = 12$ but we can recontaminate the subtree by 10 agents only, if we first clean z_3, leaving 5 agent at v. Then we clean vertex z_2 with one agent and leave this agent there. Aftre that we clean $T_x(z_1)$ with the remaining 9 agents, and finally return to z_3 for the last part.

So for unit weights we have shown that $\mathrm{cs}(T_v(z_2)) + w(z)$ are required and are also sufficient. For $\mathrm{cs}(T_v(z_1)) < \mathrm{cs}(T_v(z_2)) + w(z)$ we occupy z with $w(z)$ agents and clear all $T_v(z_i)$ by $\mathrm{cs}(T_v(z_2))$ agents but $T_v(z_1)$ last with $\mathrm{cs}(T_v(z_2)) + w(z)$ agents. $\square$

The consequence of the above Lemma is, that we can compute $\mathrm{cs}(T_v)$ in $O(n)$ time by recursively applying Equation 2.5. Alternatively, we can start from the vertices.

Exercise 13 *Compute $\mathrm{cs}(T_{v_4})$ for the tree in Figure 2.5 by the above recursive process.*

Corollary 24 *For a unit weighted tree T of size n and for a given starting vertex v we can compute the optimal monotone contiguous strategy starting at v in $O(n)$ time. An overall optimal contiguous strategy can be computed in $O(n^2)$.*

2.2.6 Optimal contiguous Intruder Search Strategy for unit weights

We consider a message based algorithm that computes the optimal number of agents required for any starting vertex v.

The following local recursive labeling $\lambda_x(e)$ for the links $e = (x, y)$ adjacent to x will be sufficient. Let $e = (x, y)$ be a link incident to x.

1. If y is a leaf, set $\lambda_x(e) := w(y)$.
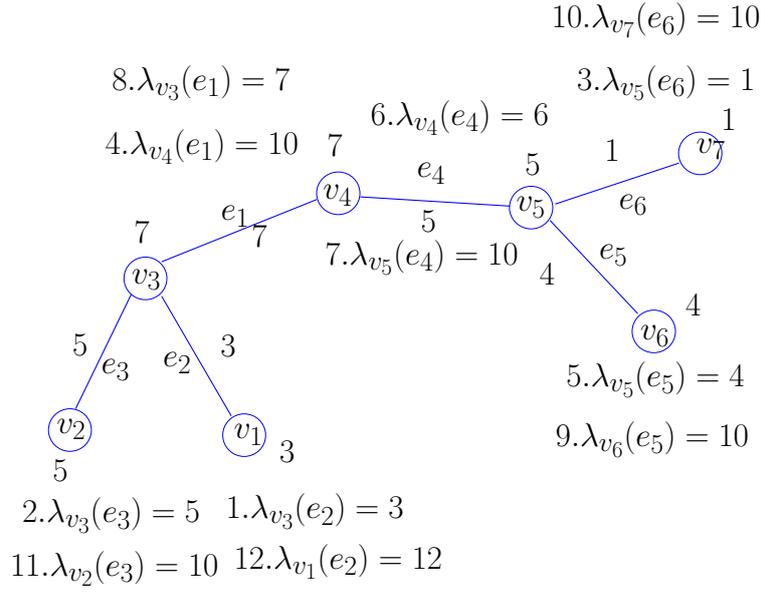
Figure 2.13: The message sending algorithm can easily work sequentially.

2. Otherwise, let d be the degree of y and let $x_1, \ldots, x_{d-1}$ be the incident vertices of y different form x. Let $\lambda_y(y, x_i) =: l_i$ and $l_i \geq l_{i+1}$. Then,

$$\lambda_x(e) := \max\{l_1, l_2 + w(y)\}.$$

For any link $e = (x, y)$ we will have two labels $\lambda_x(e)$ and $\lambda_y(e)$. By a *messages sending* technique, we can compute the labels $\lambda_x(e)$ and $\lambda_y(e)$ for all edges $e = (x, y)$ in overall linear time. Note that we interpret any link $e = (x, y)$ as undirected, which means that $(x, y) = (y, x) = e$, more formally we could have used a notion $e = \{x, y\}$.

The message sending algorithm works as follows:

1. Start with the leaves and for any leaf y and for $e = (x, y)$ send a message $l = w(y)$ to x. After receiving this messages, x sets $\lambda_x(e) := l$.

2. Consider a vertex y of degree d that has received at least $d-1$ messages l_i from the incident vertices $x_1, \ldots, x_{d-1}$ and let x be the remaining incident vertex. Let $l_i \geq l_{i+1}$. Send a message $l = \max\{l_1, l_2 + w(y)\}$ to x, after receiving the message x, set $\lambda_x((x, y)) := l$.

The above process can be applied sequentially, starting from the leaves. The process will not stop until we have send a message from x to y and y to x along any edge $e = (x, y)$. The process ends and in total $O(n)$ messages have been transmitted. An example is given in Figure 2.13. Keep in mind that setting $\lambda_x(e)$ means that x has received a message from the other vertex y of $e = (x, y)$.

Lemma 25 *The links of a tree T can be labeled with labels λ_x by the above message sending algorithm by $O(n)$ messages in total.*

Finally, we would like to prove that for an edge $e = (x, y)$ the labeling algorithm indeed computes $\mathrm{cs}(T_x(y))$ for the rooted tree T_x and its direct neighbor y. Note that we can only proof the result for unit weighted trees.

$$\mu(v_3) = \max(\lambda_{v_3}(e_1), \lambda_{v_3}(e_3) + 7) = 12$$
$$\mu(v_5) = \max(\lambda_{v_5}(e_4), \lambda_{v_5}(e_5) + 5) = 10$$

$10.\lambda_{v_7}(e_6) = 10$

$8.\lambda_{v_3}(e_1) = 7$ $3.\lambda_{v_5}(e_6) = 1$

$6.\lambda_{v_4}(e_4) = 6$

$4.\lambda_{v_4}(e_1) = 10$ 7 1 v_7

7 e_1 v_4 e_4 5 v_5 e_6

7 7 5

v_3 $7.\lambda_{v_5}(e_4) = 10$ 4 e_5

5 e_3 e_2 3 v_6 4

v_2 v_1 3 $5.\lambda_{v_5}(e_5) = 4$

5 $9.\lambda_{v_6}(e_5) = 10$

$2.\lambda_{v_3}(e_3) = 5$ $1.\lambda_{v_3}(e_2) = 3$

$11.\lambda_{v_2}(e_3) = 10$ $12.\lambda_{v_1}(e_2) = 12$
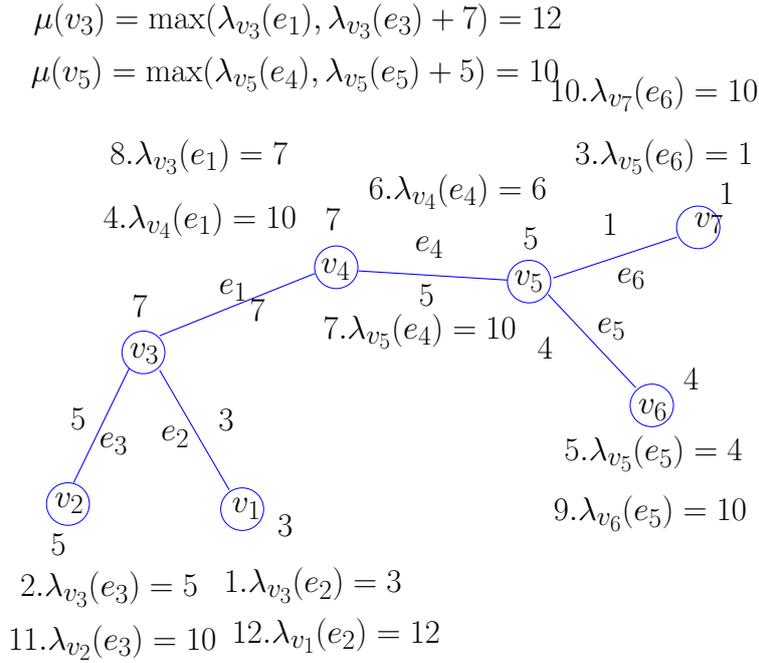
Figure 2.14: Computing $\mu(v) = \max\{\lambda_v((v, x_1)), \lambda_v((v, x_2)) + w(v)\}$ and the minimal $\min_{v \in V} \mu(v) = \mathrm{cs}(T)$ gives an optimal strategy at least for unit weighted trees.

**Lemma 26** *For a unit weighted tree $T = (V, E)$ and an edge $e = (x, y) \in E$ we have $\mathrm{cs}(T_x(y)) = \lambda_x(e)$.*

Proof. The proof goes by induction on the height $h(y)$ of $T_x(y)$. If y is a leaf we have $\lambda_x(e) = w(y)$ for $h(y) = 0$. The statement holds.

Assume that the statement holds for $0 \le h(y) < k$ and consider $h(y) = k$. For edge $e = (x, y)$ let $x_1, \ldots, x_d$ be the $d \ge 1$ be the children of y in $T_x(y)$ and assume that $\lambda_y((y, x_i)) \ge \lambda_y((y, x_{i+1}))$ holds for $i = 1, \ldots, d-1$. We also have $T_y(x_i) = \lambda_y((y, x_i))$ by induction hypothesis and $T_y(x_i) = T_x(x_i)$ by definition. Therefore we also have $\mathrm{cs}(T_x(x_i)) \ge \mathrm{cs}(T_x(x_{i+1}))$ for $i = 1, \ldots, d-1$.

In Lemma 23 the recursion Equation 2.5 for $T_x(y)$ is exactly the same as $\lambda_x((x, y))$ in step 2. of the labeling process 2.2.6. Therefore, we conclude $\mathrm{cs}(T_x(y)) = \lambda_x(y)$. $\square$

Finally, we have to compute the optimal number of agents and also a corresponding strategy. The first part can done as follows. We compute the minimum number of agents, $\mu(v)$, required for starting at a vertex v in the tree T.

For this we order all $\lambda_v((v, x_i))$ for all $i = 1, \ldots, d$ incident edges (v, x_i) so that $\lambda_v((v, x_i)) \ge \lambda_v((v, x_{i+1}))$ and compute

$$\mu(v) = \max\{\lambda_v((v, x_1)), \lambda_v((v, x_2)) + w(v)\}. \tag{2.6}$$

See for example the computation of $\mu(v_3)$ and $\mu(v_5)$ in Figure 2.14.

Altogether, we have $\mu(v) = \mathrm{cs}(T_v)$ and $\min_{v \in V} \mu(v) = \mathrm{cs}(T)$. For the movements of the agents we choose the vertex v that attains a minimum $\mu(v)$ and apply a strategy as induced by the values λ_y. We traverse T_v in increasing order of the values λ_y.

For example, in Figure 2.14 $\mu(v_5) = 10$ gives the minimal number of agents required and we start with 10 agents in v_4 w.r.t. decreasing numbers λ_{v_5}. Thus, first 1 agent moves along e_6 and back to v_5, then 4 agents move along e_5 and back to v_5. After that 10 agents move along e_4 and so on.

Theorem 27 *An optimal contiguous strategy for a unit weighted tree $T = (V, E)$ can be computed in $O(n)$ time and space.*

Proof. The number of messages required is given by the above considerations. For calculating the messages (and also the values $\mu(x)$) afterwards, we only have to register the largest three entries $\lambda_v(e)$ for any v. This can be done successively. For any new message we can adjust these largest three entries in constant time. □

2.2.7 Lower and upper bound for the contiguous search

For a given tree, T_n, with n nodes we are asking for $\max_n \mathrm{cs}(T_n) =: \mathrm{cs}(n)$. For convenience we consider the unit weighted case, where all weights are equal to 1. We will prove the following Theorem.

Theorem 28 *For unit weights and for any number of vertices n, we have $\lfloor \log_2 n \rfloor - 1 \leq \mathrm{cs}(n) \leq \lfloor \log_2 n \rfloor$.*

We prove each inequality of the Theorem separately by the following lemmata:

Lemma 29 *For every $n \geq 1$ we find trees T_n with $\mathrm{cs}(T_n) \geq \lfloor \log_2 n \rfloor - 1$.*

Proof. We consider a rooted tree T with root r and for any vertex u let the *level of u* denote the distance from r to u. If n equals $2^k - 1$ we choose a complete binary tree and show that $\mathrm{cs}(T_n) = k - 1 = \log_2(n + 1) - 1 \geq \lfloor \log_2 n \rfloor - 1$ agents are required by considering the values $\lambda_v(e)$. See also Figue 2.15.

- We have $\lambda_v((v, u)) = k - i$ and $\lambda_u((v, u)) = k - 1$, for any vertex u of level $i > 0$ and its parent node v w.r.t. r. This can be easily seen by induction. The last value stem from the fact that we have to clean a complete tree with $2^{k-1} - 1$ vertices by starting from the root node.

- We have $\mu(u) = k - 1$ for any $u \neq r$ and $\mu(r) = k$, which gives the bound.

Now, for $n \neq 2^k - 1$ consider the binary representation $n = \sum_{i=1}^r 2^{\alpha_i}$ with $\alpha_1 > \alpha_2 > \cdots > \alpha_r$. For example consider $n = 11010$ in binary representation with $\alpha_1 = 4$, $\alpha_2 = 3$, $\alpha_3 = 2$. We build a chain with vertices $x_1, x_2, \ldots, x_r$ and for any x_i we build an edge to a complete binary tree T_{α_i} of size $2^{\alpha_i} - 1$ as depicted in Figue 2.16.

This means that we have n vertices in total. We conclude that α_1 agents are required, if 2^{α_1} and 2^{α_1-1} exists in the binary representation. This holds if we start somewhere outside T_{α_1} because we visit the root of T_{α_1} at some point. If we start inside T_{α_1} (for example in a leaf) we require at most $\alpha_1 - 1$ agents for T_{α_1} but at the root node y_i of T_{α_1} we can assume that we have to place an additional agent that blocks the recontamination from x_1. For this we assume we assume that we require at least $\alpha_1 - 1 = \lambda_{y_1}((y_1, x_1))$, since the tree for 2^{α_1-1} exists. Altogether in the above case, we have $2^{\alpha_1} - 1 < n < 2^{\alpha_1+1} - 1$ and require $\mathrm{cs}(T_n) = \alpha_1 \geq \log_2(n + 1) - 1 \geq$ agents in total which gives the conclusion.

The remaining cases are as follows: Assume that 2^{α_1-1} does not exist in the binary representation, we have two subcases.

1. If all subtrees from 2^{α_1-2} to 2^0 exist, the tree starting at x_1 into x_2 is a binary tree of depth $\alpha_1 - 1$. Thus, we require again α_1 agents for any starting vertex and we can use the simple argumentation above.

$$k = 4 \text{ and } n = 2^k - 1$$
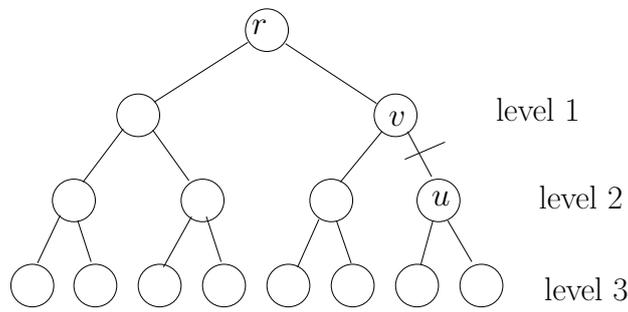
$$\lambda_v((v, u)) = k - \text{level}(u)$$
$$\lambda_u((v, u)) = k - 1$$
$$\mu(r) = k \text{ and } \mu(u \neq r) = k - 1$$

Figure 2.15: For $k = 4$ and $n = 2^k - 1$ and T_n as the full binary tree, we conclude $\text{cs}(T_n) = k - 1$ which gives the bound.

$$n = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 11010$$
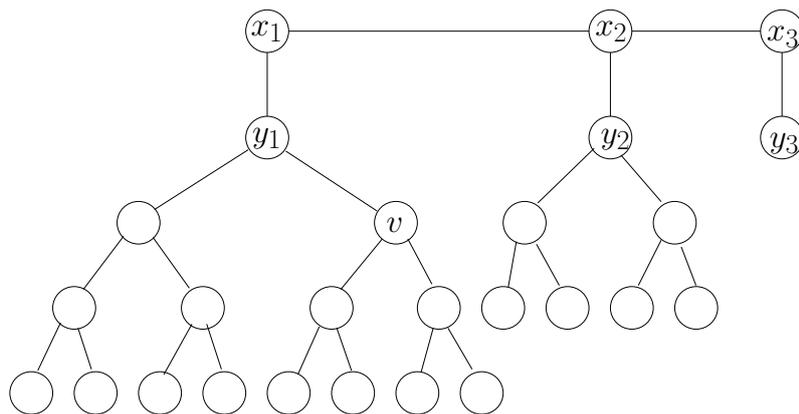
$$\lambda_{y_1}((v, y_1)) = \alpha_1 - 1$$
$$\lambda_{y_1}((x_1, y_1)) = \alpha_2 + 1 = \alpha_1$$

Figure 2.16: A tree T_n with $n = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 26$ vertices, requires $\alpha_1 = 4$ agents.

2. If not all subtrees from 2^{α_1-2} to 2^0 exist, only $\alpha_1 - 1$ agents are required. But now the value for n is small enough so that we can conclude. $\alpha_1 - 1 \geq \lfloor \log_2(\frac{2}{3}(n+1)) \rfloor \geq \log_2(n+1) - 1$ which gives the bound. This requires the measurement of $\frac{2}{3}(n+1)$ in comparison to $\alpha_1 - 1$ is left as an Exercise.

<div align="right">□</div>

Exercise 14 *Discuss the remaining case in the above proof. That is $\alpha_2 - 1 < \alpha_1$ and the two cases depicted in the proof.*

On the other hand, we show that $\lfloor \log_2 n \rfloor$ agents are always sufficient.

Lemma 30 *For every $n \geq 1$ and unit weights, $\lfloor \log_2 n \rfloor$ agents are sufficient for a contiguous search strategy.*

Proof. We consider a tree T_r with n vertices and $\mu(r) = cs(T)$. Now, we simplify this so that it becomes a complete binary tree T_r' w.r.t. r with $cs(T_r) = cs(T_r')$ by the following rules. The rules can be applied successively, until none of them is applicable any more. The children/parent relation in the tree is considered w.r.t. r.

1. For a node x and its $d > 2$ children $x_1, x_2, \ldots, x_d$ ordered by $cs(T_r(x_i)) \geq cs(T_r(x_{i+1}))$ remove all $T_r(x_i)$ for $i > 2$.

2. For a node x with two children x_1 and x_2 and $cs(T_r(x_1)) > cs(T_r(x_2))$, remove $T_r(x_2)$.

3. For a node $x \neq r$ with only one child x_1, remove x and connect x_1 to the parent of x.

4. If there are more than two vertices left, and r has only one child x_1, remove x_1 and connect the children of x_1 to r.

First, the number of agents required for T_r' and T_r are the same, because the computation of $\mu(r)$ in T_r makes use of exactly the same values. Note that the weights of the vertices are restricted to one, therefore rule 2. is also correct by $cs(T_r(x_1)) \geq cs(T_r(x_2)) + 1$. Cancelling a vertex with one child has no influence.

Second, we show that T_r' is a complete binary tree rooted in r. The first rule and the second rule returns a tree that has internal nodes with at most 2 children. Rule three deletes internal nodes with one child except for the root. Rule 4 makes the root have 2 or 0 children.

Thus, we have a binary tree whose internal nodes have degree excactly 2. Finally, we show that the tree is complete. Let x be a node such that the subtree T_x' at x is not complete and there is no other subtree in T_x' with this property. This means that the children x_1 and x_2 of x in T_r' define complete subtree T_{x_1}' and T_{x_2}' of different size. Thus, rule 2 can be applied which gives a contradiction.   □

2.2.8   The prize of connectivity

In the previous section we analyzed the contiguous search number for trees and presented a polynomial time algorithm for trees. The key argument was that recontamination does not help for decreasing the search number. The contiguous search idea is mainly based on the fact that searchers should not jump.

In general, in the non-continuous setting this is in some sense allowed. More precisely, we extend the rules defined in the beginning of Section 2.2.3. We allow that some of the agents can be retracted from somewhere in one step to be placed somewhere else in the following step.

1. Place a team of p guards on a vertex.

2. Move a team of m guards along an edge.

3. Remove a team of r guards from a vertex.

We consider the unit-weighted case in this section. Note that the monotonicity proof in the previous section also holds for non-contiguous strategies for trees and also for graphs. And it also holds, if rule 3. can be applied. This means that the progressive crusades of frontier k and the search number k for graphs correspond in general in the same way. Recontamination does not help and optimal monotone strategies always exist.

More precisely, the connectivity relationship in the proof of Lemma 19 depicted in Figure 2.7 was only shown for trees. We obtained a progressive *connected* crusade. In general, the use of a progressive crusade is sufficient. Conversely, in the proof of Lemma 20 the three cases depicted in Figure 2.10 can also be handled, if the progressive crusade is not connected.

Exercise 15 *Consider the proof of Theorem 17. Argue that with the same arguments, we can show:* For any unit-weighted graph G, with search number $s(G)$, there is always a monotone strategy with $s(G)$ searchers.

So we can ask what is the prize for the connectivity. General strategies for the above rules indeed have better search numbers as we will show here. We distinguish between the search number, $s(G)$, for general strategies and the contiguous search number, $cs(G)$, for contiguous, connected strategies. As mentioned above for both measures we always find optimal monotone strategies.

Let D_k denote a tree with root r of degree three and three full binary trees, B_{k-1}, of depth $k-1$ connected to the r. We first show that $cs(D_k) = k + 1$ holds.

Lemma 31 *For the graph D_k, we conclude $cs(D_k) = k + 1$.*

Proof. Let T_1, T_2 and T_3 denote the copies of B_{k-1} connected to the root and let e_i denote the edge that connects T_i with the root r. For the contiguous search w.l.o.g. we can assume that the edge e_1 is cleared first at timestep i_1 among the edges e_1, e_2 and e_3. W.l.o.g. let $i_2 > i_1$ denote the time step where for the first time a leaf l of T_2 or T_3 is reached. Assume w.l.o.g. $f \in T_2$. At time step $i_2 - 1$ the path $P(r, x_k) = r, x_1, \ldots, x_{k-1}$ of length $k - 1$ from r to the neighbor x_{k-1} of f with k vertices has to be clean and for any $x_i \in P(r, f)$ there is a unique subtree in T_2 different from f that is not fully decontaminated. For the root r there is a subtree in T_3 that is not fully decontaminated. So at least one searcher for any x_i and for r is required which gives k in total. One additional searcher now is required for cleaning f. This gives $cs(D_k) \geq k + 1$.

On the other hand, $k + 1$ searchers are sufficient, if we start at the root with $k + 1$ agents and clean the trees T_i successively with k agents. Recursively, a full binary subtree of depth l is cleaned from the root with $l + 1$ searchers. □

Now, we would like to relate this to the number $s(D_k)$. We consider D_{2k-1} with $cs(D_{2k-1}) = 2k$.

Lemma 32 *For D_{2k-1} we conclude $s(D_{2k-1}) \leq k + 1$.*

Proof. For $k = 1$ the statement is trivial. So assume $k > 1$. We first place one agent at the root r and successively clean the copies of B_{2k-2} by k agents. The last statement is shown by induction. For $k = 2$ we clean a subtree of depth one by two agents. Move via the root to the other subtree of depth one. Then we move back to the root.
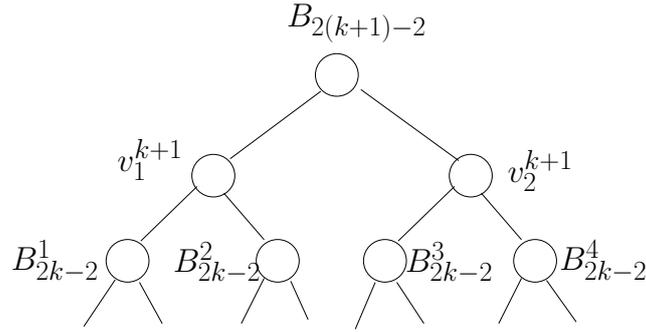
Figure 2.17: The inductive step. Each subtree B_{2k-2}^i can be cleaned by k agents. Placing an additional searcher at v_1^{k+1} in the beginning, we first clean B_{2k-2}^1 by k agents and B_{2k-2}^2 successively by k agents, move all $k+1$ searchers to v_2^{k+1} and do the same for B_{2k-2}^3 and B_{2k-2}^4 successively. Altogether, $k+1$ searchers are sufficient for $B_{2(k+1)-2}$.

Let us assume that the statement holds for $k \geq 2$. We can fully clean B_{2k-2} with k searchers that are finally located at the root. The tree $B_{2(k+1)-2}$ has four subtrees B_{2k-2}^i for $i = 1, 2, 3, 4$ of depth $2k - 2$ as shown in Figure 2.17 that can be cleaned by induction hypothesis with k agents. Let v_1^{k+1} be the ancestor of B_{2k-2}^1 and B_{2k-2}^2 and let v_2^{k+1} be the ancestor of B_{2k-2}^3 and B_{2k-2}^4.

For cleaning any B_{2k-2}^i for $i = 1, 2, 3, 4$ we have to take care that while *crossing* the root vertex of B_{2k-2}^i with k agents (which is the transition of cleaning one child of B_{2k-2}^i after the other), we have to avoid recontaminations that can come from the edge between v_j^{k+1} and B_{2k-2}^i. We realize this by cleaning the edge from B_{2k-2}^i to v_j^{k+1} with one agent, while $k-1$ agents stay in B_{2k-2}^i. After that we resume the cleaning the other child of B_{2k-2}^i.

So we first place one additional agent at v_1^{k+1} and clean B_{2k-2}^1 and B_{2k-2}^2 successively by k agents with the special intermediate step explained above. Then all $k+1$ agents move via the root toward v_2^{k+1}. Here we again leave the additional agent at v_2^{k+1} and clean B_{2k-2}^3 and B_{2k-2}^4 successively by k agents as before. Thus $k+1$ agents are required in total.

By induction, B_{2k-2} can be cleaned by k searchers and for D_{2k-1} at most $k+1$ searchers are required. $\qquad\square$

Now, we have a fixed relationship between $cs(G)$ and $s(G)$ for $G = D_{2k-1}$. We have $s(D_{2k-1}) \leq k+1$ and $cs(D_{2k-1}) = 2k$.

Corollary 33 *There exists a tree T so that $cs(T) \leq 2s(T) - 2$ holds.*

It was also shown by Barrière et al. 2012, that there is no tree T with ratio $\frac{cs(T)}{s(T)}$ larger than 2. More precisely,

$$\frac{cs(T)}{s(T)} < 2 \text{ for all trees } T.$$

The proof of this fact relies on the fact that in principle (up to retractions) the trees D_k can be considered to be the category of graphs that gives the worst-case ratio. If there is some time left at the end of the semester we will prove this fact.

Chapter 3

Discrete Cop and Robber Game

In this chapter we would like to discuss another discrete variant of the intruder search problem. In comparison to the previous chapter, we assume that at any time of the game the position of the single intruder is given.

More precisely, there is a single robber R and a set of cops C and a graph $G = (V, E)$. The game starts with the cops, by choosing the starting vertices for the set C. After that, the robber R can choose its starting vertex. The game runs in subsequent steps. First, any cop can move from a vertex to an incident vertex, then the robber can move. The game ends, when one cop enters the position of the robber or the robber enters the position of a cop, respectively.

Cop and Robber game for graphs:
Instance: A Graph $G = (V, E)$ and the cardinality of the cops C.
Question: Is there a winning strategy S for the cops C?

We are searching for classifications of graphs that allow a winning strategy for C or vice versa a winning strategy for R. Aigner and Fromme introduced the problem in the midst of the 90ies.

3.1 Classifications of graphs

3.1.1 Simple examples and pitfalls

It is interesting to see that it makes a difference, if we do not allow the robber to keep in place during its strategy. This is called the *active* version of the game, in correspondance to the *passive* version, where the robber is not forced to move in any step.

Figure 3.1 shows an example where this makes a difference for a single cop. In the active version the cop starts at vertex v_1 and the robber can only choose the opposite vertex r_2. The cop moves toward v. Now the robber has to move to r_1. The cop moves toward v_2 and after the next mandatory move of the robber, the robber will be caught. In the passive version the robber can move around or rest in the 4-cycle and holds distance 2 from the cop all the time. In the following we will always discuss the more intuitive passive version of the game. Let G_C denote the set of all graphs that allow a winning strategy for C and let G_R denote all graphs that have a winning strategy for R.

Obviously, any tree T belongs to G_C already for a single cop, that successively moves into the subtree of R. Additionally, for a single cop, all graphs that contain a cycle of length at least 4 belong to G_R.

We concentrate on a single cop. In the winning case for the cop, the final situation is as follows: The robber is located in a vertex v_r and the cop is located in v_c for an edge $e = (v_r, v_c)$. Moreover, all neighbors, $N(v_r)$, of v_r are also neighbors of v_c, which means $N(v_r) \subseteq N(v_c)$.
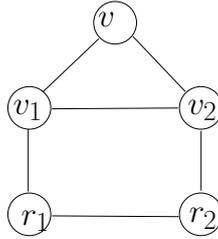
Figure 3.1: In this simple graph for one cop and a robber it makes a difference, if the robber has to perform moves mandatorily.
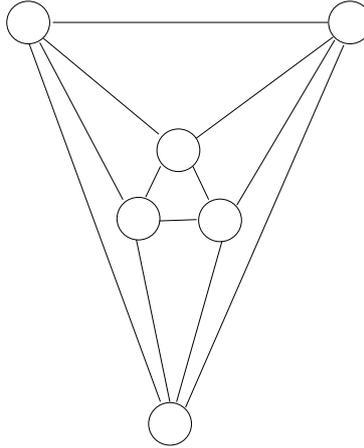
Figure 3.2: A graph without pitfalls.

For a pair (v_r, v_c) of vertices we call v_r a *pitfall* and v_c its *dominating vertex* if $N(v_r) \cup \{v_r\} \subseteq N(v_c)$ holds. Obviously, a graph G whithout a pitfall is in G_R. Figure 3.2 shows an example.

Exercise 16 *Present a construction scheme for graphs of arbitrary size without pitfalls.*

3.1.2   Algorithmic approaches

We would like to show that for a single cop the classification of a graph depends on the successive removement of pitfalls of G.

Lemma 34 *Let v_r be a pitfall of some graph G. Then*

$$G \in G_C \iff G \setminus \{v_r\} \in G_C \,,$$

where $G \setminus \{v_r\}$ results from G by removing all edges adjacent to v_r and vertex v_r from G.

Proof. If $G \setminus \{v_r\} \in G_R$ holds, the robber simply identifies any visit of the cop of the pitfall v_r by the dominating vertex v_c and makes use of a strategy in $G \in G_R \setminus \{v_r\}$. Note that the neighborship of the pitfall v_r is a subset of the neighorship of v_c. This means that we do not *weaken* the cop by the identification.

If $G \setminus \{v_r\} \in G_C$ holds, the cop wants to extend its winning strategy to G. The cop simply identifies any visit of the robber of the pitfall v_r as a visit of the dominating vertex v_c and makes use of the same strategy. Again, since the neighborship of the pitfall v_r is a subset of the neighorship of v_c, we do not weaken the robber by this identification. If at the end we catch the

imaginary robber at v_c and the robber is physically located at v_r, we will catch the robber in the next step. $\qquad\square$

Now, we have a simple characterization of G_C.

Theorem 35 *The graph G is in G_C, if and only if the successive removement of pitfalls finally ends in a single vertex. The classification of a graph can be computed in polynomial time.*

Proof. Lemma 34 gives the key argument, as the classification does not change by removing pitfalls. This means that we either end up in a graph with no pitfalls for $G \in G_R$ or in a single vertex for $G \in G_C$.

Checking the existance of a pitfall can be done locally for any vertex and its neighborship. After computing the neigborship sets, we can check the pitfall property for a vertex in a brute-force manner in $O(n^2)$ time and for all vertices in $O(n^3)$ time for a graph with n vertices. At most n reduction steps can be done. $\qquad\square$

Exercise 17 *Design an efficient algorithm for checking the pitfall property of a single vertex and/or for the graph.*

The above shrinking process answers the classification question algorithmically in polynomial time. On the other hand we would like to construct arbitrary examples of representatives of G_C. It can be shown that G_C is closed under the operations *product of two graphs* and *reduction of a graph*.

The *product* $G_1 \times G_2$ of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, G_2)$ is defined by vertex set $V_1 \times V_2$ and an edge set by the folllowing rules: (v_1, v_2) and (w_1, w_2) of $V_1 \times V_2$ build an edge if:

1. $v_1 = w_1$ and $(v_2, w_2) \in E_2$ or

2. $(v_1, w_1) \in E_1$ and $v_2 = w_2$ or

3. $(v_1, w_1) \in E_1$ and $(v_2, w_2) \in E_2$.

Lemma 36 *If $G_1, G_2 \in G_C$, then $G_1 \times G_2 \in G_C$*

Proof. If the cop has a winning strategy for G_1 that starts in v_1^s and catches the robber in v_1^e and for G_2 that starts in v_2^s and catches the robber in v_2^e, the cop can start in (v_1^s, v_2^s) apply the strategies simultaneously and finally catches the robber in a vertex (v_1^e, v_2^e). This strategy is obviously correct. $\qquad\square$

For a graph G and its subgraph H, the *retraction* from G to H is a mapping $\varphi : V(G) \mapsto V(H)$ of the vertices of $V(G)$ of G to the vertices $V(H)$ of H as follows: $\varphi(H) = H$ for $(u, v) \in E$ we have $(\varphi(v), \varphi(u)) \in E(H)$. The graph H is called a *retract* of G, if a retraction from G to H exists.

Note that $G \setminus \{v_r\}$ for a pitfall v_r is a retract of G.

Lemma 37 *If $G \in G_C$, and graph H is a retract of G, then $H \in G_C$.*

Proof. Assume that $H \in G_R$ holds and let φ be the mapping for a retraction from G to H. We would like to show $G \in G_R$. We extend the winning strategy for H to a winning strategy of G as follows: R remains in H and identifies the moves of C in G as moves in H. That is, if C moves from v to u in G, the robber indentifies this move as a move from $\varphi(u)$ to $\varphi(w)$ which

exists in H by definition of φ. The robber always moves according to the strategy in H and cannot be caught. $\square$

Note that, the above lemmata do not rely on the fact that there is only one cop.

Theorem 38 *The class of graphs G in G_C is closed under the operations product and retraction.*

3.1.3   How many cops are required?

Obviously, any graph with a 4-cycle will not belong to G_C, therefore it makes sense to think about more than one cop. For a graph G the *cop-number*, $c(G)$, denotes the minimum number of cops required to guarantee that $G \in G_C$ holds.

A *vertex cover* of a graph G is a subset $V_c \subseteq V$ so that any vertex $u \in V \setminus V_c$ has a neighbor in V_c. Therefore the minimum vertex cover is an upper bound on $c(G)$. First, we show that $c(G)$ can be arbitrarily large for some graphs.

Theorem 39 *Let $G = (V, E)$ be a graph with minimum degree n that contains neither 3- nor 4-cycles. We conclude $c(G) \geq n$.*

Proof. Let us assume that $n-1$ cops are sufficient. If G does not have a vertex cover of size smaller than n, the $n-1$ cops located in the beginning at $c_1, \ldots, c_{n-1}$ cannot prevent the robber to choose a safe vertex. So the robber chooses such a vertex whose neighbors are not occupied by the cops. Since there are no 3- and 4-cycles, by the next move a single cop cannot threaten (occupy and/or be adjacent to) two neighbors of the robber in the next step. Therefore, there is still one safe neighbor for the robber after the next move of the cops.

It remains to show that a vertex cover of size $< n$ does not exist. Consider any vertex set $V = \{v_1, \ldots, v_{n-1}\}$ of G and a vertex $w \neq v_i$ for $i = 1, \ldots, n-1$. Note that $|V| \geq n$ holds, so w exists. Now consider the neighborhood, $N(w)$, of w. Let it consist of k vertices $v_1, \ldots, v_k$ from V and $l - k$ vertices $w_1, \ldots, w_{l-k}$ not in V. We have $l \geq n$, $k \leq n-1$ and $l - k \geq 1$. There are no 3- and 4-cycles, so $N(w_i) \cap N(w_j)$ has to be $\{w\}$ for $i \neq j$. If the set V is a vertex cover for G, any $N(w_i)$ has to contain a different vertex from V. But none of the $N(w_i)$'s can contain a vertex of $v_1, \ldots, v_k$, since this would give a 3-cycle with w. This means that we require $l - k$ different vertices from $v_{k+1}, \ldots, v_{n-1}$ and n vertices from V in total, a contradiction. $\square$

We can construct regular graphs of arbitrary size, which fulfill the condition of Theorem 39. The following Theorem is given by construction.

Theorem 40 *For every n there exists a graph without 3- or 4-cylces with minimum degree n. So, for any n there is a graph with $c(G) \geq n$.*

Proof. For $n = 2$ the simple 5-cycle will work. Note that C_5 is 3-colorable, which means that we color the vertices by three colora such that no two colors are adjacent. Three colors are required and sufficient for C_5. Inductively, we construct a 3-colorable graph with degree exactly n for any vertex v of G and without 3- and 4-cycles. The coloring is required for maintaining the cycle condition by construction.

The induction base for $n = 2$ was shown above. Let us assume that the statement holds for n. We consider four copies G_0, G_1, G_2, G_3 of a corresponding graph G for n as depicted in Figure 3.3. We build new edges with respect to the coloring of the vertices so that any vertex obtains an additional edge; see Figure 3.3. From G_i to G_j the exact copies of two vertices of a single color are connected. For example from G_1 to G_2 all identical copies of color 3 are
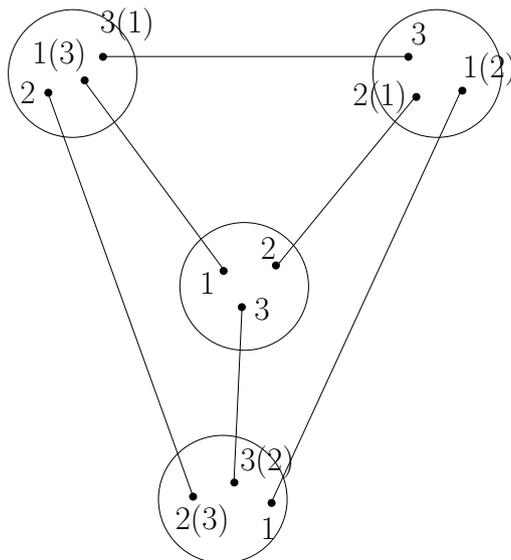
Figure 3.3: In the inductive step we use four copies G_0, G_1, G_2, G_3 of a 3-colorable graph G of degree exactly n without 3- and 4-cycles. Then we construct new edges according to the colors and finally interchange some colors, appropriately.

connected, from G_2 to G_0 all identical copies of color 2 are connected and so on. There is a unique correspondance as shown in Figure 3.3.

Since there are no cycles of size 3 or 4 in G_0, G_1, G_2, G_3 and any two edges between G_i and G_j make use of identical copies of the same color there are at least two edges between them in G_i and G_j respectively. So there are no 3- or 4-cycles in the new graph. For the inductive step, we require a *new* 3-coloring, which will be attained by interchanging the colors for example color 3 by 1 and color 1 by 3 in G_1 and color 2 by 1 and color 1 by 2 in G_2 and so on. Thus, we maintain 3-coloring in G_0, G_1, G_3, G_4 and also for the connections. □

Finally, in this section we prove some positive results by bounding the cop-number from above for special graphs. The corresponding proofs are constructive, i.e., a winning strategy for the cops can be computed.

Theorem 41 *Consider a graph G with maximum degree 3 and the property that any two adjacent edges are contained in a cycle of length at most 5. Then $c(G) \leq 3$.*

Proof. The proof is constructive in the following sense. If the position of the robber is known, for the cops c_1, c_2 and c_3 we consider three paths towards r that use all incident edges to r. We choose P_1, P_2 and P_3 for c_1, c_2 and c_3 respectively. The paths cover the incident edges by different cops and with length l_1, l_2 and l_3. And the paths make use of any possible shortcut for reaching the incident edges. Note that the paths need not be disjoint and r might also have only one or two incident vertices. But such paths do always exist. We would like to argue that by the condition of the Theorem, we can decrease the overall distance $l := l_1 + l_2 + l_3$ in any move of the cops.

Formally, after the move of the robber, R, we move c_1, c_2 and c_3 to c_1', c_2' and c_3' so that $l' < l$ holds. We further assume that r was adjacent to exactly three vertices r_1, r_2 and r_3. The other cases can be handled anlogously, and are given as an exercise. We have $P_1 = \{c_1, \ldots, r_1, r\}$, $P_2 = \{c_2, \ldots, r_2, r\}$ and $P_3 = \{c_3, \ldots, r_3, r\}$ and consider the following cases.

1. The robber R stands still. The cops move along the paths toward R and $l' \leq l - 3$.
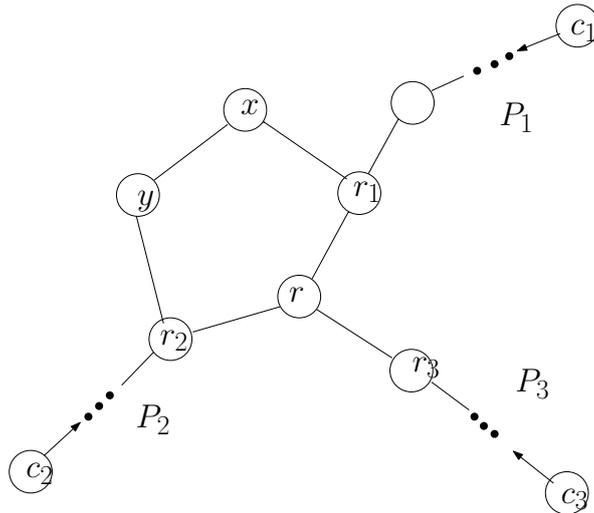
Figure 3.4: If r has degree 3 and c_1 is not on r_1, there is a 5-cycle so that we can move closer to r at least by one.

2. The robber R moves to r_1 (w.l.o.g.):

   r_1 **has degree 1:** This cannot happen since (r, r_1) and (r, r_2) are neighboring edges.

   r_1 **has degree 2:** Either c_1 was on r_1 and we are done or move all three cops toward r which gives $l' \leq l_1 - 2 + l_2 + l_3 = l - 2 < l$.

   r_1 **has degree 3:** Either c_1 was on r_1 and we are done or we have $l_1 \geq 2$. At least one adjacent vertex, say x, of r_1 does not belong to P_1, otherwise we use a shortcut for P_1. This means that (x, r_1) and (r_1, r) are on a cycle of length at most 5. Since r has only degree 3, one of the vertices r_2 or r_3, (say r_2) also belong to this cycle as depicted in Figure 3.4. So we have a 5-cycle r_2, y, x, r_1, r. We move all three cops toward r, respectively r_1 and use the paths $P_1 = \{c_1', \dots, r_1\}$ $P_2 = \{c_2', \dots, r_2, y, x, r_1\}$ and $P_3 = \{c_3', \dots, r_3, r, r_1\}$ with length $l' \leq l_1 - 2 + l_2 + 1 + l_3 = l - 1 < l'$.

In any case we can reduce the distance of the cops to the robber for the paths to all all three neighboring vertices.   □

Finally, we would like to prove that for any planar graph G, indeed $c(G) \leq 3$ holds. We first show that in any graph G it is always possible to protect a shortest path between two vertices by two cops. Protection means the robber cannot enter the path without being caught in the next step. The path length is given by the number of edges along a path between to vertices in G. By this measure the triangle inequality holds.

Lemma 42 *Consider a graph G and a shortest path $P = s, v_1, v_2, \dots, v_n, t$ between two vertices s and t in G, assume that we have two cops. After a finite number of moves the path is protected by the cops so that after a visit of the robber R of a vertex of P the robber will be caught.*

Proof. First, we move a cop c onto some vertex $c = v_i$ of P. Let $d(x, y)$ denote the distance between two vertices in G. The robber R located at r can only have a shorter distance to vertices on one side of P w.r.t. c because the triangle inequality holds. Assuming, that r is closer to some x in $s, v_1, \dots, v_{i-1}$ *and* some y in $v_{i+1}, \dots, v_n, t$ is a contradiction to the shortest path between x and y. That is $d(x, c) + d(y, c) \leq d(x, r) + d(r, y)$. This means that $d(r, x) < d(c, x)$ only holds at most for one side of P w.r.t. c and for the other side we conclude $d(r, y) > d(c, y)$ in this case.
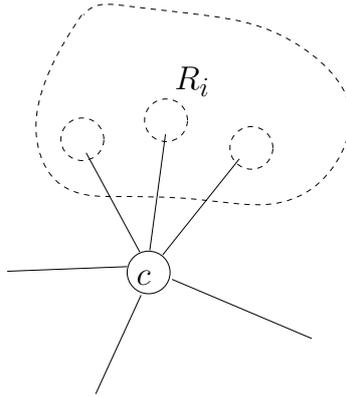
Figure 3.5: Case 1: All three cops in one vertex.

Thus, we move c towards the vertices x. Now the robber can move. Again, if there are still vertices on one side of P w.r.t. c which are closer to r than to c we move further on toward these vertices. So finally, we achieved

$$d(r, v) \geq d(c, v) \text{ for all } v \in P \tag{3.1}$$

by this process.

Now the robber again could make its move. We show that we can maintain the inequality at all time, which means that the robber will be caught if it tries to move towards the vertices of P.

Assume Equation 3.1 holds. The robber can either stay in its place, so the cop c does and we fulfill Equation 3.1 (and the second cop could move now). Or the robber moves and tries to contradict Equation 3.1 by its single move. Assume r goes to some vertex r', we have

$$d(r', v) \geq d(r, v) - 1 \geq d(c, v) - 1 \text{ for all } v \in P.$$

If there is again some $v' \in P$ with $d(c, v') - 1 = d(r', v')$, we have the same situation as above and we can move c toward v' and Equation 3.1 holds again. Again as before the movement toward r' cannot reduce the distance to x and y on opposite sides of c w.r.t. P. Thus, by the move toward some v' we fulfill Equation 3.1. □

Finally, we exploit the above property for planar graphs and by the use of 3 cops and two such paths.

Theorem 43 *For any planar graph G we have $c(G) \leq 3$.*

Proof. We show that the region R_i for the robber R will shrink successively, that is $R_{i+1} \subset R_i$ after some moves of the cops. Two situations can appear.

Case 1: All three cops occupy a single vertex c and the robber is located in one component R_i of $G \setminus \{c\}$; see Figure 3.5.

Case 2: There are two different paths P_1 and P_2 from v_1 to v_2 that are protected in the sense of Lemma 42 by cops c_1 and c_2; see Figure 3.6. In this case $P_1 \cup P_2$ subdivides G into an interior, I, and an exterior region E. That is $G \setminus (P_1 \cup P_2)$ has at least two components. W.l.o.g. we assume that R is located in the exterior $E = R_i$.

We will show that these two cases can appear successively and the region R_i of the robber will shrink. In the beginning all cops are located in a single vertex c and we are in case 1. We show how we handle the cases.
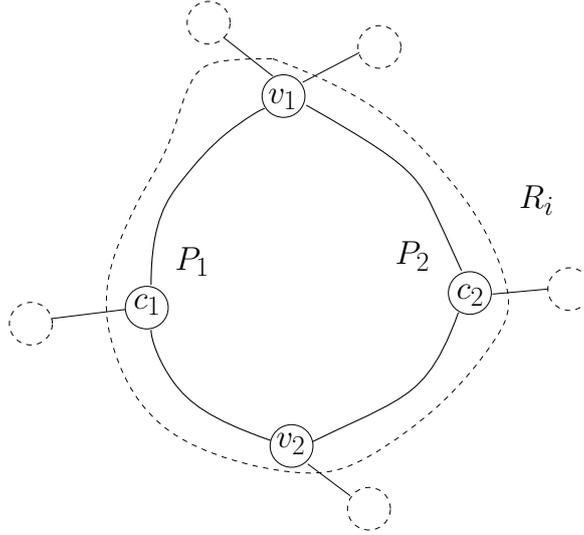
Figure 3.6: Case 2: Two cops protect two paths.

Movements in Case 1: We consider different situations for the neighbors of c:

c **has one neighbor in R_i:** Move all cops to this neighbor c' and consider $R_{i+1} = R_i \setminus \{c'\}$. This gives Case 1 again.

c **has more than one neighbor in R_i:** Let a and b be two of the neighbors and let $R(a,b)$ be a shortest path in R_i between a and b. One cop remains in c, another cop protects the path $P(a,b)$ by Lemma 42. Thus $P_1 = a, c, b$ and $P_2 = P(a,b)$. We are in Case 2 with $R_{i+1} \subset R_i$.

Movements for Case 2: We consider the situation for the composition of R_i and the location of the robber. We first assume that there is another shortest path $P'(v_1, v_2)$ but different from P_1 and P_2 that partially runs in R_i and also connects v_1 and v_2.

Let x_1 be the first vertex of $P'(v_1, v_2)$ where the path P' leaves $P_1 \cup P_2$ after v_1 and x_2 be the first vertex of P' after x_1 that enters $P_1 \cup P_2$ again.

We let c_3 protect the path P_3 which results from combining $P_{1,2}(v_1, x_1)$ with $x_1, r_1, \ldots, r_l, x_2$ and with $P_{1,2}(x_2, v_2)$ as depicted in Figure 3.7. While c_1 and c_2 protect P_1 and P_2, the cop c_3 can protect this path P_3 from v_1 to v_2. At the end c_3 protects P_3 and c_1 or c_2 protect the remaining path, we are in Case 2 with $R_{i+1} \subset R_i$.

On the other hand, if there is no path different from P_1 and P_2 and partially running in R_i that connects v_1 and v_2, there are no such leave and entry vertices x_1 and x_2 that are connected in R_i. Thus, the robber has to be inside a component that fully is connected to a single vertex x on P_1 and P_2. Thus, we move c_3 to this vertex, and also c_1 and c_2 and end in Case 1 again. $\square$
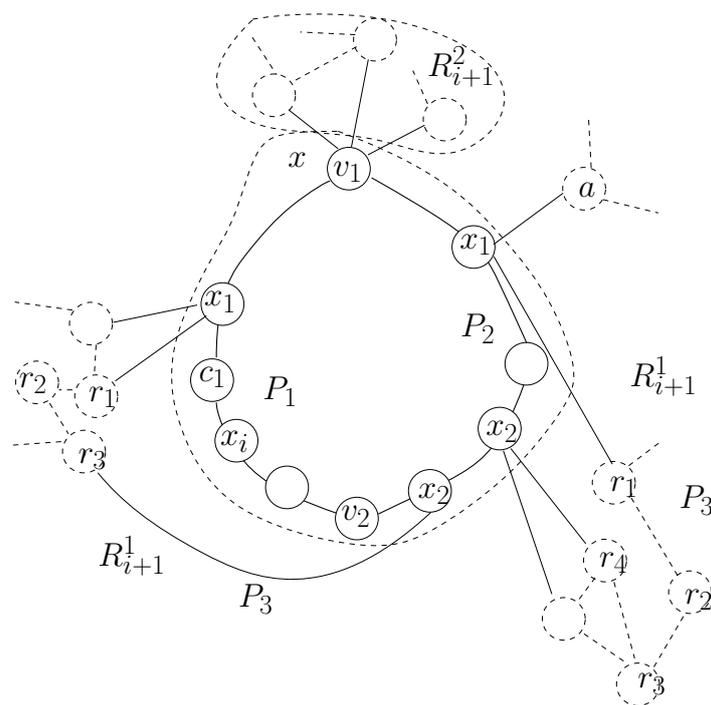
Figure 3.7: The situation for the two chains P_1 and P_2 protected by c_1 and c_2. If there is another shortest path from v_1 to v_2 different from P_1 and P_2 that runs partially in R_i, we construct a path P_3 that runs from v_1 to v_2 that is protected by c_3 alone and protects vertices of R_i. If there is no such path, a vetex x exists that can be visited by all cops and gives Case 1 again.

Chapter 4

Randomized Variants

In this chapter and before turning over to some geometric variants of the intruder search problem, we would like to resume the graph decontamination problem for stationary guards in order to show that there are also randomized strategies and problem variants that can be discussed.

We show a slightly better approximation as the greedy algorithm for trees by a randomized strategy. Additionally, we interpret the search number of a graph in the configuration that the fire spreads from any vertex with the same probability. We concentrate on positive results.

4.1 Better approximations for trees by randomization

We pick up the firefighter problem for trees again. As already asked for in Exercise 10 we can formulate the problem as an integer LP by the following rules. Let $v \leq u$ denote that v equals u or is a predecessor of u w.r.t. the root r of tree T.

$$
\begin{array}{rll}
\text{Minimize} & \sum_{v \in V} x_v w_v & \\
\text{so that} & x_r = 0 \;=\; 0 & \\[2mm]
& \sum_{v \leq u} x_v \;\leq\; 1 & : \quad \text{for every leaf } u \\[2mm]
& \sum_{v \in L_i} x_v \;\leq\; 1 & : \quad \text{for every level } L_i, i \geq 1 \\[2mm]
& x_v \;\in\; \{0, 1\} & : \quad \forall\, v \in V
\end{array}
$$

In the above integer LP the weights w_v denote the number of vertices in the subtree T_v of vertex v w.r.t. the root r.

Let opt_{ILP} denote the optimal solution of the above integer LP. For the approximation we solve the problem in polynomial time for $x_v \in \mathbb{R}^{\geq 0}$. The optimal solution, opt_{RLP}, is a fractional solution so that a subtree T_v with $x_v = a \leq 1$ is called *a-saved*, a portion $a \cdot w_v$ of the subtree is saved. For two vertices v_1 and v_2 on the same path from the root r to a leaf u and v_1 is ancestor of v_2 and $x_{v_1} = a_1$ and $x_{v_2} = a_2$ the vertices of T_{v_2} are $(a_1 + a_2)$-saved. The remaining vertices of T_{v_1} are only a_1-saved.

The simple idea is that we would like to use a rounding scheme. But we do this in a randomized fashion. For each level we interpret the a-values as a probability dsitribution for choosing a vertex to be safe. This is a rounded strategy w.r.t. the distribution. On each level we simply choose a vertex at random according to its distribution. Note that the sum of the a-values for level i could be smaller than 1. We interpret the remaining portion as the probability of choosing none of the vertices in this level. The main problem is that we might choose vertices that are

on the same path from the root to a leaf. If no such *double-protections* occur the expected value of the rounded strategy would be at least opt_{ILP} and the expected approximation value would be indeed 1.

If also a successor of a vertex is choosen by our procedure, we simply delete it in the final solution and do not choose another vertex at this level. This makes the choosing procedure independent for every level. Altogether, the only loss we have is for the double-protections. Let us assume that they can occur. What happens if the a tree T_{v_i} at level i is *fully* saved by the fractional strategy? We would like to argue that in the worst-case the fractional strategy has assigned a $1/i$ fraction to all vertices on the path from r to v_i and the subtree is fully saved by the rounding scheme with probability

$$1 - (1 - 1/i)^i \geq 1 - \frac{1}{e} \, .$$

We put this intuition into a formal argument.

Theorem 44 *Consider an algorithm that protects the vertices w.r.t. the probability distribution given by* opt_{RLP}*. The expected approximation ratio of the above strategy for the number of vertices protected is* $\left(1 - \frac{1}{e}\right)$*.*

Proof. Let S_F denote the fractional solution for opt_{RLP}. For an integer solution, we choose a vertex on each level w.r.t. the probability distribution from opt_{RLP}. Let S_I denote the outcome of this assignment. We would like to show, that the expected value of S_I is larger than $\left(1 - \frac{1}{e}\right)$ times the value of S_F which in turn outperforms opt_{ILP}.

Let x_v^F denote the value of x_v for the fractional strategy and let x_v^I denote the value $\{0,1\}$ of the integer strategy. For convenience we denote $y_v = \sum_{u \leq v} x_u \in \{0,1\}$, which indicates whether v is finally saved or not. Let $y_v^F = \sum_{u \leq v} x_u^F \leq 1$ denote the fraction of v saved by the fractional strategy. For $y_v = 1$ it suffices that one of the predecessor of v was chosen. Let $r = v_0, v_1, v_2, \ldots, v_k = v$ be the path from r to v, so we have

$$\mathbf{Pr}[y_v = 1] = 1 - \prod_{i=1}^{k}(1 - x_{v_i}^F) \, .$$

For example, the probability that v_2 is safe is $x_1 + (1 - x_1)x_2 = 1 - (1 - x_1)(1 - x_2)$ and the probability that v_3 is safe is $1 - (1-x_1)(1-x_2) + (1-x_1)(1-x_2)x_3 = 1 - (1-x_1)(1-x_2)(1-x_3)$ and so on.

Thus we compute

$$
\begin{aligned}
\mathbf{Pr}[y_v = 1] &= 1 - \prod_{i=1}^{k}(1 - x_{v_i}^F) \\[2mm]
&\geq 1 - \left(\frac{\sum_{i=1}^{k}(1 - x_{v_i}^F)}{k}\right)^k = 1 - \left(\frac{k - \sum_{i=1}^{k} x_{v_i}^F}{k}\right)^k \\[2mm]
&= 1 - \left(\frac{k - y_v^F}{k}\right)^k \\[2mm]
&= 1 - \left(1 - \frac{y_v^F}{k}\right)^k \geq 1 - e^{-y_v^F} \geq \left(1 - \frac{1}{e}\right) y_v^F \, .
\end{aligned}
\tag{4.1}
$$

The first inequality is a standard inequality for means of positive real values $\frac{x_1 + x_2 + \cdots + x_n}{n} \geq \sqrt[n]{x_1 \cdot x_2 \cdots x_n}$. The second and third inequalities stem from classical analysis where we use the fact that $0 \leq y_v^F \leq 1$ holds.

The value of S_F is simply the sum of all y_v^F. Thus, we conclude

$$\mathbf{E}(|S_I|) = \sum_{v \in V} \mathbf{Pr}[y_v = 1] \geq \left(1 - \frac{1}{e}\right) \sum_{v \in V} y_v^F = \left(1 - \frac{1}{e}\right)|S_F|.$$

$\square$

Altogether, we have a randomized polynomial time algorithm for trees with an expected approximation ratio better than the $\frac{1}{2}$-approximation of greedy.

Exercise 18 *Prove $\frac{x_1 + x_2 + \cdots + x_n}{n} \geq \sqrt[n]{x_1 \cdot x_2 \cdots x_n}$ for positive real values x_i. Also prove the last two inequalities of Equation 4.1 in the proof of Theorem 44.*

4.2 Search numbers for random fire sources

The second part on randomization is that we might use it to consider the situation that the starting vertex of the fire has some influence on the number of agents required. Therefore, in this section we again consider the firefigther problem on graphs but the start of the fire is choosen uniformly at random among all vertices. The question is, what is the number of agents required so that for a given class C of graphs it can be expected that at least linear number of vertices can be saved.

This subsumes many questions handled before. We would like to have a classification by the properties of C, we would like to find a minimum number k of agents required and we use an expected value for assuming that the fire can start in any vertex with the same probability.

For a graph $G = (V, E)$ and a fixed number k of agents, the k-surviving rate, $s_k(G)$, is the expectation of the *proportion* of vertices that can be saved if the fire can start from any vertex with the same probability. We are looking for classes, C, of graphs G so that for a fixed constant ϵ, $s_k(G) \geq \epsilon$ holds for any $G \in C$. This means that at least $\epsilon \cdot |V|$ vertices will be saved. For a given graph G, a given k and a vertex $v \in V$ let $\text{sn}_k(G, v)$, denote the number of vertices that can be protected by k agents, if the fire starts at v.

We are also searching for the minimal number k that guarantees $s_k(G) \geq \epsilon$. This means that

$$\frac{1}{|V|} \sum_{v \in V} \text{sn}_k(G, v) \geq \epsilon |V|$$

has to be shown. For a class C let the minimum number k that guarantees $s_k(G) > \epsilon$ for any $G \in C$ be denoted as the firefighter-number, $\text{ffn}(C)$, of C.

Firefighter-Number for a class C of graphs:
Instance: A class C of graphs $G = (V, E)$.
Question: Assume that the fire breaks out at any vertex of a graph $G \in C$ with the same probability. Compute $\text{ffn}(C)$.

Theorem 45 *For planar graphs we have $2 \leq \text{ffn}(C) \leq 4$.*

There is a simple argument for the lower bound $2 \leq \text{ffn}(C)$. Consider a planar bipartite complete graph with 2 and $n - 2$ vertices on the corresponding sides. For any starting vertex at most one vertex can be saved and $\frac{1}{n}$ will become arbitrarily small.

For the upper bound we first show a somewhat easier result that shows the main idea. The vertices are subdivided into classes X and Y, where a root vertex from set X allows to save many (a linear number of) vertices and a root vertex from the set Y allows to save only few (almost zero) vertices. Finally, $|Y| \leq c|X|$ gives the bound.

Theorem 46 *For planar graphs G with no 3- and 4-cycle, we have $s_2(G) \geq 1/22$.*

Proof. We make use of the Euler formula, $c + 1 = v - e + f$, for planar graphs with e edges, v vertices, f faces and c components. We assume that the graph is connected, that is $c = 1$. A planar graph with no 3- and 4-cycle has average degree less than $\frac{10}{3}$. If not, we assume $\frac{10}{3}v \leq 2e$, summing up the degrees of all vertices gives exactly twice the number of edges and we count less than that by assuption. We can also conclude $5f \leq 2e$, since any face has at least 5 edges that can neighbor two faces, but we might not count all edges twice. This means $f \leq \frac{2}{5}e$. Inserting $v \leq \frac{3}{5}e$ into the formula gives $f \geq 2 + \frac{2}{5}e$, a contradiction. With similar arguments we can show that a graph with no 3-, 4 and 5-cylces has average degree less than three, which is the question of Exercise 19.

We subdivide the vertices V of G into groups w.r.t. the degree and the neighborship.

- Let X_2 denote the vertices of degree ≤ 2.

- Let Y_4 denote the vertices of degree ≥ 4.

- Let X_3 denote the vertices of degree exactly 3 but with at least one neighbor of degree ≤ 3.

- Let Y_3 denote the vertices of degree exacly 3 but with all neighbors having degree > 3 (degree 3 vertices not in X_3).

Let x_2, x_3, y_3 and y_4 denote the cardinality of the sets, respectively.

Let $|V| = n$. For a vertex starting in X_2, by two agents we protect the neighbors and safe $n - 2$ vertices. For a vertex in X_3, we save two neighbors so that the fire spreads to the neighbor u of degree ≤ 3 and in the next step we protect the remaining neighbors of u, thus protecting $n - 2$ vertices in total. For starting vertices in Y_3 and Y_4, we assume that we can save no vertex.

We have to show that $\frac{1}{n} \sum_{v \in V} \mathrm{sn}_k(G, v) \geq \epsilon \cdot n$ holds and we consider

$$s_2(G) = \frac{1}{n^2} \sum_{v \in V} \mathrm{sn}_k(G, v) \geq \frac{1}{n^2}(x_2 + x_3)(n - 2) = \frac{n - 2}{n} \cdot \frac{x_2 + x_3}{x_2 + x_3 + y_3 + y_4} \qquad (4.2)$$

since $x_2 + x_3 + y_3 + y_4 = n$ holds.

We first would like to compute a correspondance between Y_3 and Y_4 and consider the graph $G_Y = (V_Y, E_Y)$ that consists of the edges of G with precisely one vertex in Y_3 and one vertex in Y_4. The graph G_Y has precisely $3y_3$ edges and at most $y_3 + y_4$ vertices. Note that some of the vertices of Y_4 might be neighbors for more than one vertex of Y_3. The graph G_Y is bipartite and a subgraph of G. A cycle of size 5 has to go forth and back from Y_3 to Y_4 vertices and has to end at the same class Y_4 or Y_3. Therefore in G_Y we only have cycles of size at least 6 and by Exercise 19 the average degree of vertices of G_Y is at most 3. This means by counting $3(y_3 + y_4)$, we have counted at least any edge twice, which gives $3(y_3 + y_4) \geq 6y_3$ and $y_3 \leq y_4$.

Now we would like to compute a fixed relation between $x_2 + x_3$ and $y_3 + y_4$. By the average degree, by counting $\frac{10}{3}(x_2 + x_3 + y_3 + y_4)$ edges we have at least counted $3x_3 + 3y_3 + 4y_4$ edges, which gives $9x_3 + 9y_3 + 12y_4 \leq 10(x_2 + x_3 + y_3 + y_4)$ and in turn $2y_4 - y_3 \leq 10x_2 + x_3$. By $y_3 \leq y_4$ we have $y_4 \leq 10x_2 + x_3$ and also $y_3 + y_4 \leq 20x_2 + 2x_3 \leq 20(x_2 + x_3)$.

Now insertion into Equation 4.2 gives

$$s_2(G) \geq \frac{n - 2}{n} \cdot \frac{x_2 + x_3}{x_2 + x_3 + y_3 + y_4} \geq \frac{n - 2}{n} \cdot \frac{x_2 + x_3}{21(x_2 + x_3)} = \frac{n - 2}{21n} . \qquad (4.3)$$

If G has only two vertices, in any case the vertex distinct from the root can be saved. If G has $3 \leq n \leq 44$ vertices, at least $\frac{2}{44}$ are saved in a single step. For $n \geq 44$ we have $s_2(G) \geq \frac{42}{21 \cdot 44} = \frac{1}{22}$. So the expected value of saved vertices is always $\frac{1}{22}n$. $\square$

Exercise 19 *Prove by the Euler formula that a graph with no 3-, 4-cycle and 5-cylces has average degree less than three.*

Finally, we would like to prove the statement $\text{ffn}(C) \leq 4$ of Theorem 45. To this end we prove the following result with a precise value of $s_4(G)$ for planar graphs.

Theorem 47 *Using four firefighters in the first step and then always three firefighters in each step, for every planar graph G there is a strategy such that $s_4(G) \geq \frac{1}{2712}$ holds.*

Proof. We can assume that G is a maximal planar graph without multi-edges. Inserting more edges will only help the fire but multi-edges will not. This means that G is a triangulation and we can assume that any face has exactly three edges.

We provide the proof in several steps and the proof contains four Lemmate, namely Lemma 48, Lemma 49, Lemma 50, and Lemma 51.

Similarily, to the proof above we subdivide the vertices V of G into sets X and Y. Where X will be the set of vertices where a strategy saves at least $n - 6$ vertices and for Y we do not expect to save any vertex, for $|V| = n$.

The final conclusion is that for some $\alpha = \frac{1}{872}$ we will conclude

$$|Y| \leq \left(93 + \frac{3}{\alpha}\right)|X| = 2709|X|. \tag{4.4}$$

Thus from $|X| + |Y| = n$ and Equation 4.4 we conclude

$$s_4(G) \geq \frac{n-6}{n} \cdot \frac{|X|}{|X| + |Y|} > \frac{n-2}{n} \cdot \frac{|X|}{2710|X|} = \frac{n-6}{2710n}. \tag{4.5}$$

For $n \geq 10846$ we have

$$s_4(G) \geq \frac{1}{2710} - \frac{6}{4 \cdot 2710^2} \geq \frac{2710 - 3/2}{2710^2} \geq \frac{1}{2712}$$

For $2 \leq n < 10846$ we save at least $\min(4, n-1)$ in the first step, which gives also $s_4(G) \geq \frac{1}{2712}$.

The remaining task is, to establish the above bounds. First, we subdivide the vertices accordingly. Note that for starting vertices of degree 3 or four we can save $n - 1$ vertices in the first step.

- For degree $3 \leq d \leq 6$ let X_d denote the vertices that guarantee to save at least $|V| - 6$ vertices.

- All other vertices form the set Y_d for $d \geq 5$.

Also note that a starting vertex v of degree 5 with a neighbor u of degree at most 6 is in X_5. Because of the triangulation u and v have two common neighbors n_1 and n_2. In the first step, we let the fire only spread to u by protecting 4 neigbors at u. Then the neighbors v, n_1 and n_2 of u are already protected. So we fully protect the graph in the next step by 3 agents.

We require some more structural properties for the relationsship between X and Y which stem from the triangulation. The length of a path in the graph is given by the number of edges.

Lemma 48 *For a vertex $v \in Y_6$ there is a path of length at most 3 from v to a vertex u that has degree distinct from v (i.e., $\neq 6$) and the inner vertices of the path have degree exactly 6.*
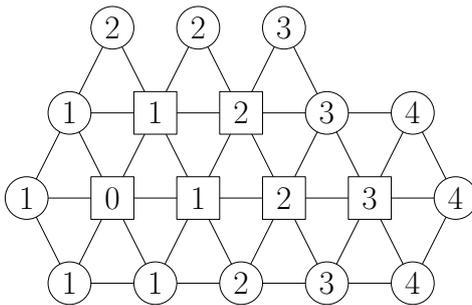
Figure 4.1: If a vertex v of degree 6 is in Y_6, we will find a vertex u as given in Lemma 48 in the neighborhood, or we finally end in a situation where v is blocked locally in a hexagon and thus belongs to X_6.

Proof. Let us assume that this is not the case. In the first step we can always protect 4 subsequent neighbors of v as depicted in Figure 4.1. If one of the remaining two neighbors (step 1) does not have degree 6, we are done. So assume that also these two neigbors have degree 6. Because of the triangulation, they will span a hexagon and we can protect 3 neighbors of these two as depicted in Figure 4.1. The fire spread to only two remaining neighbors (step 2). If one of them does not have degree 6 we are done again. So assume that both also have degree 6 and we extend the hexagonal grid. We can protect the neighbors by three agents as depicted in Figure 4.1 and only one vertex remains on fire after the fire spreads (step 3). If this vertex does not have degree 6 we are done again. But if this vertex also have degree 6 we will finally enclose the fire in the next step and only 6 vertices (v, 2 in (step 1), 2 in (step 2), 1 in (step3)) gets burned in total, a contradiction to $v \in Y_6$. Without the above property, v will be in X_5! □

The next lemma tells us something about vertices from Y_d with $d \geq 7$ related to y_5. Let $d(v)$ denote the degree of vertex v.

Lemma 49 *A vertex with $d(v) \geq 7$ has at most $\lfloor \frac{1}{2}d(v) \rfloor$ neighbors in Y_5.*

Proof. A neighbor u of v from Y_5 has two neighbors n_1 and n_2 in common with v. If one of them has degree strictly less than 7, the vertex u belongs to X_5. So the vertices from Y_5 around v are *seperated* by vertices of degree ≥ 7, which gives the bound.   □

Finally, we make use of the following structural lemma that stems from the Euler formula and the simple, maximal planar triangulation.

Lemma 50 *For a simple, maximal planar graph we have*

$$\sum_{v \in V}(d(v) - 6) = -12 \,. \tag{4.6}$$

Proof. For a maximal, simple planar graph we have $3f = 2e$, by counting the edges of every triangular face, we count any edge exactly twice. Additionally, we have $\sum_{v \in V} d(v) = 2e$ because summing up the degree of the vertices counts any edge twice as well. The Euler formula says $v - e + f = 2$ and we conclude $v - e + \frac{2}{3}e = 2 \iff 2e - 6v = -12$ which gives the conclusion. □

Now we present the main idea for obtaining Equation 4.4. The idea is that we distribute the *intitial potential* $p_1(v) := (d(v) - 6)$ of every vertex among the others so that finally any vertex has potential $p_2(v)$ and also $\sum_{v \in V} p_1(v) = \sum_{v \in V} p_2(v) = -12$ holds.

The rules for the distribution are as follows:

Rule A: A vertex v of degree at least 7 gives a value of $\frac{1}{4}$ to each neighbor vertex from Y_5.

Rule B: For a vertex $v \in Y_6$ we choose exactly one vertex u with $d(u) \neq 6$ and distance $d(v, u) \leq 3$ as in Lemma 48. The vertex u gives a value of $\alpha > 0$ to v.

We would like to choose α accordingly, the property $\sum_{v \in V} p_1(v) = \sum_{v \in V} p_2(v) = -12$ will hold since the distribution is cost neutral by construction. Such a distribution with desired properties exists.

Lemma 51 *There is a constant $\alpha > 0$ such that a distribution by Rule A and B gives $\sum_{v \in V} p_1(v) = \sum_{v \in V} p_2(v) = -12$ and for every $v \in X$ we have $p_2(v) \geq -3 - 93\alpha$ and for every $v \in Y$ we have $p_2(v) \geq \alpha$.*

Before we prove this final lemma, we use its conclusion. An $\alpha = \frac{1}{872}$ will do the job. We then conclude

$$-12 = \sum_{v \in V} p_2(v) \geq (-3 - 93\alpha)|X| + \alpha|Y|$$

and this gives

$$|Y| \leq \left(93 + \frac{3}{\alpha}\right)|X| < 2790|X|$$

which is Equation 4.4. It remains to prove Lemma 51.

Proof of Lemma 51.

For the claim we estimate the distribution of the potential as given by the rules A and B.

Considering Rule B, how often can a vertex u with $d(u) \neq 6$ give a potential of α to some vertex v? We give a rough upper bound with respect to the maximal distance ≤ 3 from u:

- Distance 1: $d(v)$ times to a direct neighbor, if all of them are in Y_6. This gives $1 \cdot d(u)$.

- Distance 2: For all $d(v)$ neighbors of the first case, at most 5 times, if the $d(v)$ neighbors of the above case have degree 6 and all 5 remaining neigbors are from Y_6. This gives $5 \cdot d(u)$.

- Distance 3: For all vertices of the second case, at most 5 times, if the vertices of the second case all have degree 6 and the remaining neighbors are from Y_6. This gives $25 \cdot d(u)$.

Altogether, any vertex u with $d(u) \neq 6$ can give a potential α at most $(1 + 5 + 25)d(u) = 31d(u)$ times. This gives upper bounds for the potential $p_2(v)$:

- $v \in X_3$: We have $p_2(v) \geq -3 - 93\alpha$ because $d(v) = 3$ and $p_1(v) = -3$.

- $v \in X_4$: We have $p_2(v) \geq -2 - 124\alpha$ because $d(v) = 4$ and $p_1(v) = -2$.

- $v \in X_5$: We have $p_2(v) \geq -1 - 155\alpha$ because $d(v) = 5$ and $p_1(v) = -1$.

For vertices of degree 6 we have the following:

- $v \in X_6$: $p_2(v) = 0$ because $d(v) = 6$ and $p_1(v) = 0$.

- $v \in Y_6$: $p_2(v) = p_1(v) + \alpha = \alpha$ because Rule B gives a single value α from some u to v, and by Lemma 48 such a vertex exists, if v exists.

Note that the distributions of these α potentials are cost-neutral in total for p_1 and p_2.

Considering Rule A, for vertex v and $d(v) \geq 7$, we can apply Lemma 49 and the above estimate for an upper bound

$$p_2(v) \geq (d(v) - 6) - \left\lfloor \frac{1}{2}d(v) \right\rfloor \cdot \frac{1}{4} - 31d(v)\alpha. \tag{4.7}$$

So the remaining cases can be estimated by

- $v \in X \cup Y$ with $d(v) = 7$: $p_2(v) \geq \frac{1}{4} - 217\alpha$.

- $v \in X \cup Y$ with $d(v) \geq 8$: $p_2(v) \geq d(v)\left(\frac{7}{8} - 31\alpha\right) - 6$ by $\lfloor \frac{1}{2}d(v) \rfloor \cdot \frac{1}{4} \leq \frac{1}{8}d(v)$.

We choose $\alpha > 0$ so that $\frac{1}{4} - 217\alpha \geq \alpha$ and $d\left(\frac{7}{8} - 31\alpha\right) - 6 \geq \alpha$ holds for all $d \geq 8$. The first inequality is fulfilled for $\alpha = \frac{1}{218\cdot 4} = \frac{1}{872}$ and this also fulfills the second inequality. Thus, we have shown that for $v \in Y$ $p_2(v) \geq \alpha$ holds. The same value for α also guarantess $p_2(v) \geq -3 - 93\alpha$ for all $v \in X$.

It remains to note that the distribution is cost-neutral also for Rule A. But this is clear because the sum of the potential added and retracted is the same. Altogether, Lemma 51 holds and the full conclusion can be drawn, which is $s_4(G) \geq \frac{1}{2712}$.   □

Exercise 20 *Present the precise strategies that stem from Theorem 46 and Theorem 47. Analyse the corresponding running time.*

Chapter 5

Geometric Firefighting

In this chapter we would like to discuss geometric fireighting settings in the Euclidean plane. We assume that there is a fire source s (a point in the plane) and the fire spreads in all directions with the same unit speed 1. This means that the fire can be considered to be an expanding circle, expanding with speed one over time.

The adversary of the fire expansions can build barriers with some speed b. In the geometric setting, a barrier is a curve in the plane with given start and endpoint. We consider different models. For example, we can assume that the barriers have to be build up successively one after the other or in parallel. We can also restrict the type of the barrier, barriers can be restricted to line segment or can be arbitrary curves. Additionally, we consider different types of environments.

5.1 Firefighting in a simple polygon

Let us assume that the fire spreads inside a simple polygon P at a single point s. Let us further assume that a set of m potential line segment barriers, $B = \{b_1, b_2, \ldots, b_m\}$, is given. Each barrier, b_i, has a start point and an end point t_i on the boundary of P.

All barriers can be build with the same speed b. If $|b_i|$ is the length of the barrier, its *relative completion time* is $\frac{|b_i|}{b}$. We also assume that the barriers have to be constructed one after the other. During the construction of the barrier the fire should never reach the current end point while the barrier is constructed. Any barrier saves a certain amount of the total area of the polygon. The goal of the barrier construction is, that we would like to find a schedule for a valid construction of the barriers, that saves a maximal portion of the total area of the polygon. An example for a polygon, a fire source s and a set of potential barriers is shown in Figure 5.1.

In contrast to the Intruder-Definition given in Section 1.1.1 we define the problem as a firefighter problems in a simple polygon.

Geometric Firefighter Problem in a simple polygon
Instance: A simple polygon, a fire that spreads from a given starting point $s \in P$ with speed 1, a set of m line segment *barriers*, $b_1, b_2, \ldots, b_m$, which can be successively constructed with the same speed B.
Output: Compute a valid sequence of barriers that is constructed successively so that the area protected from the fire is maximized.

As already shown in Theorem 1 in Section 1.1.1 this problem is NP-hard in general. It can also be shown that it is approximation hard, which means that there is no polynomial time approximation scheme. Therefore we have to consider constant approximations for the optimal schedule.

5.2   A general approximation algorithm

In this section we describe a general approach for the approximation of a scheduling problem. We also relate it to the above firefighting problem.

We consider m jobs $b_1, b_2, \ldots, b_m$ with precise latest starting time and duration. That is, the job b_i has to be started before time s_i and has a duration of d_i.

For a *valid schedule* of jobs after n processing steps of an algorithm with l_n entries, we have a consecutive list of jobs $J_n = (b_{n_1}, b_{n_2}, \ldots, b_{n_{l_n}})$ which can be scheduled one after the other. Let $s'_{n_k} \leq s_{n_k}$ be the precise starting time of the scheduled jobs. We require $\sum_{k=1}^{j} s'_{n_k} + d_{n_k} \leq s_{n_{j+1}}$ for $j = 1$ to $l_n - 1$. The next job ha to be started before its starting option ends. This means that the jobs can be started one after the other without conflicts of the starting times.

Each job b_i will contribute with a profit A_i to an overall profit A. Profit A_i will be attained, if the job is scheduled. These profits might *overlap* for different jobs. Thus, the profit of a job b_i may change over time, if another job b_j was scheduled before that also covers parts of the profit that can be covered by b_i.

For a current schedule of jobs J_n we denote the current profit of a job $b_j \notin J_n$ by $A_j(J_n)$. This depends on the overlapping. More precisely for $J_n = (b_{n_1}, b_{n_2}, \ldots, b_{n_{l_n}})$ and $b_j \notin J_n$ we define

$$A_j(J_n) := A_j \setminus \left(\bigcup_{b_{n_k} \in J_n} A_{n_k} \right).$$

(Note that here A_{n_k} is the initial overall potential profit of $b_{n_k} \in J_n$ but this profit is always covered by the profit of J_n, even though some of the profit A_{n_k} was intermediately already covered by another b_l in J_n.)

We start with an empty schedule J_0 and successively process one of the remaining jobs. This job will never be processed any more after that. So for m jobs we are done after m processing steps.

For the scheduling algorithm and a current schedule J_n after n steps we sort all non-rejected and non-scheduled jobs by decreasing relative profits $\frac{A_j(J_n)}{d_j}$. The profit of the job is considered in relation to its duration. We apply the following *GlobalGreedy* procedure to the current job b_j with maximal $\frac{A_j(J_n)}{d_j}$:

1. If a job b_j with current largest value $\frac{A_j(J_n)}{d_j}$ can be scheduled somewhere in J_n it is inserted into J_n for obtaining J_{n+1}.

2. If the job cannot be scheduled because some of the jobs overlap with its possible execution interval, we check whether we can delete such (consecutive) jobs out of J_n with the following property: For a given constant $\mu < 1$, the profit of the deleted jobs is smaller than μ times the profit $A_j(J_n)$ of job b_j. Additionally, after deleting the jobs, job b_j can be scheduled. All jobs that have been deleted will never be considered again. We build J_{n+1} by inserting b_j to the remaining list of J_n. There might be some options for such sequences.

3. If such a sequence of jobs does not exist, job b_j is rejected and will never be considered again. $J_{n+1} := J_n$.

For convenience for the profits in the procedure and also for the jobs we define corresponding colors. In the beginning all potential profits are colored red. Profits and jobs that will be scheduled by rule 1. or rule 2. are colored green. A profit (and a job) that was colored green in the schedule and is rejected later by rule 2. will be colored grey. For a schedule J_n let J'_n
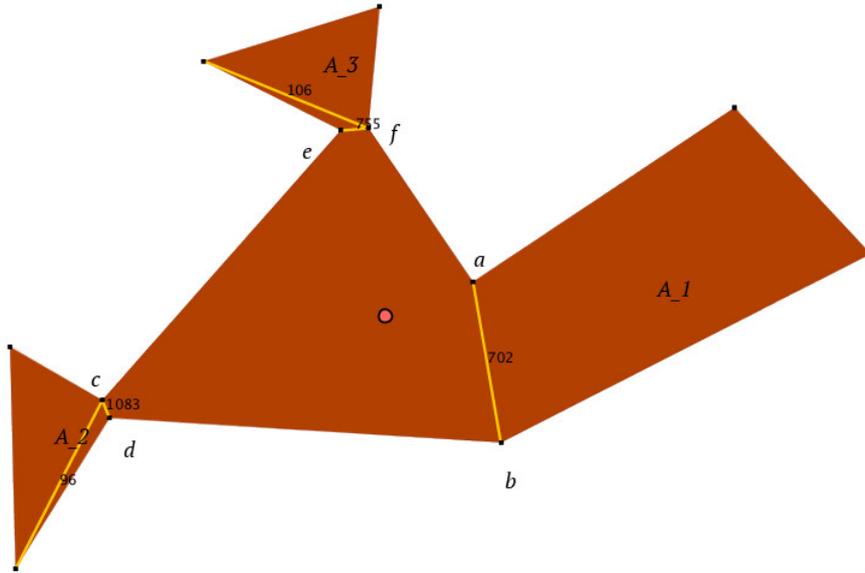
Figure 5.1: Considering the three barriers $b_1 = (a, b)$, $b_2 = (c, d)$ and $b_3 = (e, f)$ with $A_1 = 1053$, $A_2 = 162.45$ and $A_3 = 188.75$ and speed $b = 2$. We first the barriers by priority $p_1 = 702 = 2A_1/|b_1|$, $p_2 = 1083 = 2A_2/|b_2|$ and $p_3 = 755 = 2A_3/|b_3|$. The intermediate schedule $J_2 = (b_2, b_3)$ blocks the construction of b_1. Since $\mu \cdot A_1 > A_3$ and $(|b_1| + |b_3|)/2 < d(s, a)$ holds we delete b_3 and insert b_1.

denote the list of all jobs that have been inserted during the construction of J_n. The set J'_n also contains the deleted barriers of color grey and the current barriers of color green.

The above algorithm is a general scheme for a corresponding scheduling problem. We would like to apply it to our geometric firefighter problem and first give a very simple example. In Figure 5.1 there is a fire source and a set of potential barriers which are diagonals.

In this setting for convenience we require that the barrier has to be fully constructed before the fire reaches any point of it. This requirement is not necessary in general but by considering the closest point from b_i to s and the duration d_i of the barrier we can easily compute the a constraint for b_i. In general the fire should not have reached the current construction point of the barrier.

We consider only the barriers $b_1 = (a, b)$, $b_2 = (c, d)$ and $b_3 = (e, f)$ as depicted in Figure 5.1, where each barrier will be started with its first vertex. We assume that the building speed is $b = 2$. Each barrier subdivides the polygon into two parts and potentially would save the part A_i that does not contain the fire source. Let $|b_i|$ denote the lenght of each b_i, we assume $|b_1| = 3$, $|b_2| = 0.3$ and $|b_3| = 0.5$. The distance $d_P(s, a)$ from s to a in P is assumed to be 1.8. We further have calculated $A_1 = 1053$, $A_2 = 162.45$ and $A_3 = 188.75$ which are the profits of the barriers in the beginning. For the scheduling algorithm we order them relatively by priority $p_i = \frac{2A_i}{|b_i|}$ because $d_i = \frac{|b_i|}{2}$ is the duration for constructing b_i. The values in the figure denote the corresponding priority values p_i This means that $p_1 = 702$, $p_2 = 1083$ and $p_3 = 755$ holds. We assume that $\mu = 0.2$.

The scheduling algorithm first chooses b_2 and b_3 in this order because of the priorities and of the fact that both barriers can be indeed scheduled in this order. The fire will not reach c and e after $\frac{0.3+0.5}{2} = 0.4$ time steps. Because $(|b_1| + |b_2| + |b_3|)/2 = 1.9$ exceeds the distance from s to a in P, in the third step of the process we would like to apply rule 2. Since $\mu \cdot A_1 > A_3$ holds, we delete b_3 out of the schedule and build b_1 instead. The final schedule is $J_3 = (b_2, b_1)$ which saves A_2 and A_1. The resulting corresponding color scheme is shown in Figure 5.2.
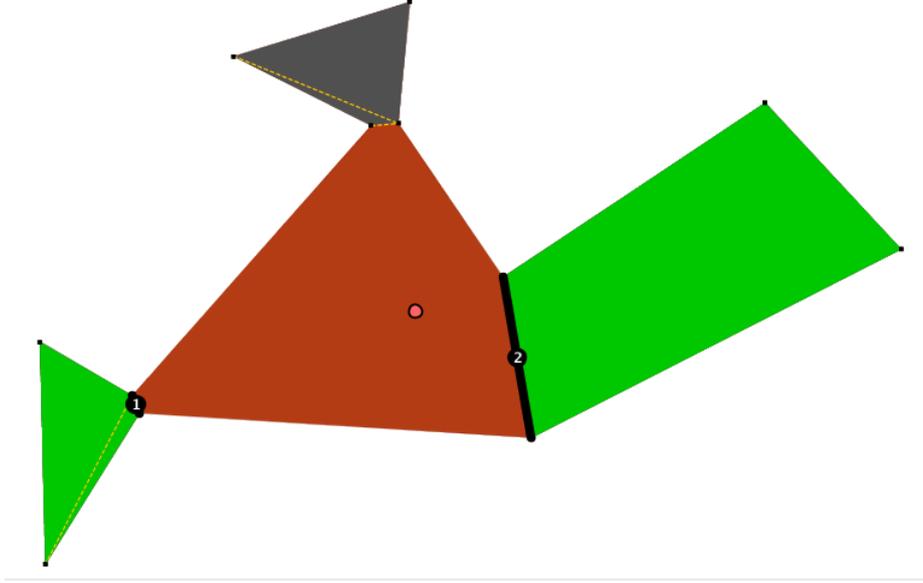
Figure 5.2: The color scheme of the scheduling algorithm for Figure 5.1. After b_1 exchanges b_3 the profit A_3 is colored grey. The part that was not protected remains red-colored. The barriers are build in the order 1. and 2. as depicted.

The following Lemma relates the green profit with the grey profit, so that we do not loose to much by deletions. By rule 2. a profit gets color grey, if it was deleted for insertion of a green profit larger than μ times the grey profit. Therefore we can prove the following relationship. Let J_m denote the final schedule for a set of jobs $b_1, \ldots, b_m$ and let $J_n(grey)$ and $J_n(green)$ denote the profits that are colored green and grey during the construction of J_n.

Lemma 52 *By the* GlobalGreedy *procedure we finally have*

$$J_m(grey) \leq \frac{\mu}{1-\mu} J_m(green) . \tag{5.1}$$

Proof. By induction on number of jobs processed during GlobalGreedy. The lemma holds in the beginning for the empty schedule J_0. Let us assume that the lemma holds after n steps for J_n. Consider step $n + 1$.

If no job is deleted by rule 1. or rule 3. only the green profit will increase from J_n to J_{n+1} that is

$$J_n(grey) = J_{n+1}(grey) \leq \frac{\mu}{1-\mu} J_n(green) \leq \frac{\mu}{1-\mu} J_{n+1}(grey) .$$

If by rule 2. some of the jobs where cancelled out of J_n for inserting b_j in the $n + 1$ step. The overall profit of the cancelled jobs (that becomes grey now) is smaller than μ times $A_j(J_n)$, the profit of b_j right now. We conclude

$$
\begin{aligned}
\frac{\mu}{1-\mu} J_{n+1}(green) &\geq \frac{\mu}{1-\mu}(J_n(green) + (1-\mu)A_j(J_n)) \\
&\geq \frac{\mu}{1-\mu} J_n(green) + \mu A_j(J_n) \geq J_n(grey) + \mu A_j(J_n) \geq J_{n+1}(grey),
\end{aligned}
$$

where we used the induction hypothesis for the secon $\geq$ sign in the second line. $\square$

For the relationsship between the profit of the final schedule J_m and an optimal schedule J_{opt} we make use of another color, blue.

For the profits, all profit elements in the optimal schedule J_{opt} that are still have color red after the application of J_m will be colored blue. So none of the grey and green profits of J_m become blue. The blue profit part depends on J_m and J_{opt}, let us denote this profit, by $J_m(blue)$. It is clear that

$$|J_{\text{opt}}| \leq J_m(blue) + J_m(green) + J_m(grey) .$$

holds, we would like to express $J_m(blue)$ in terms of $J_m(green)$ and $J_m(grey)$.

We can assume that the jobs of J_m are colored green and grey by the schedule. Likewise, we assign the blue property to the jobs of J_{opt}. The blue property is assigned to the first job of J_{opt} that covers a portion of the blue profit. Such a job cannot be green or grey, because otherwise it cannot participate at a the red colored profit at the end of the schedule J_m. Thus, all three color classes of jobs are pairwise disjoint.

As already mentioned J'_m denote the list of all jobs that have been inserted during the construction of J_m. Any green and grey jobs of J'_m has obtained a unique execution time, that will not be changed by the process afterwards (although grey jobs will no longer be scheduled at the end).

We make use of a scheme that pays money from green and grey jobs to the blue jobs w.r.t. the execution intervals of the jobs $b_i \in J'_m$ and $b_j \in J_{\text{opt}}$.

1. If the execution interval of $b_j \in J_{\text{opt}}$ is fully included in the execution interval of $b_i \in J'_m$, the job b_i pays its green or grey profit times $\frac{d_j}{d_i} < 1$ to b_j.

2. If the execution interval of $b_j \in J_{\text{opt}}$ overlaps with the execution interval of $b_i \in J'_m$, the job b_i pays its green or grey profit times $\frac{1}{\mu}$ to b_j.

The payment of each green and grey job to the blue jobs has an upper bound w.r.t. their profit.

Lemma 53 *Any single green or grey job from J'_m pays in total at most $1 + \frac{2}{\mu}$ times its profit to the blue jobs.*

Proof. A job $b_i \in J'_m$ has a fixed execution interval I_i and the start- and endtime of the interval can only be located inside at most two intervals of jobs from J_{opt} which are also executed successively. This gives at most 2 times $\frac{1}{\mu}$ the profit of b_i. For all jobs from $b_j \in J_{\text{opt}}$ fully inside the execution interval I_i of b_i a portion $\frac{d_j}{d_i}$ of b_i's profit is payed for $d_j < d_i$. In total this can only sum up to at most 1.   □

The other way round, any blue job is payed by at least its own profit from the green and grey jobs.

Lemma 54 *Any single blue job from J_{opt} achieves at least a payment in the size of its blue profit from the green and grey jobs.*

Proof. We consider a blue job $b_j \in J_{\text{opt}}$. This job was rejected at some step $k + 1$ and let $J_k = (b_{k_1}, b_{k_2}, \ldots, b_{k_{l_k}})$ be the corresponding schedule of the algorithm. For the final exection time interval of $b_j \in J_{\text{opt}}$ consider an consecutive subset $\overline{J_k}$ of the jobs of J_k of mimimal size whose intervals overlap with the execution time of b_j, The deletion of $\overline{J_k}$ would allows to schedule b_j.

The total sum profit of the jobs of $\overline{J_k}$ is larger than μ times the current profit $A_j(J_k)$ of b_j which is currently red. Some of this red part $A_j(J_k)$ will become blue at then end but may be not all of it. In any case μ times the current red part $A_j(J_k)$ of b_j is larger than μ times the final blue

part of b_j. On the other hand, the job b_j had priority less than all previously inserted jobs of J_k which is $\frac{A_i(J_k)}{d_i} \geq \frac{A_j(J_k)}{d_j}$.

We consider the size of $\overline{J_k}$. If $|\overline{J_k}| = 1$ holds for the single job, say b_i, the execution time of $b_j \in J_{\text{opt}}$ might be fully inside the execution time of b_i, which gives

$$A_i(J_k)\frac{d_j}{d_i} \geq A_j(J_k)\frac{d_i}{d_i} = A_j(J_k)$$

where $A_i(J_k)$ will be the green or grey profit of b_i for the final J_m and $A_j(J_k)$ is larger than the blue profit of b_j.

For $|\overline{J_k}| \geq 1$, the execution time interval of b_j is not fully inside an execution interval of a job in $\overline{J_k}$, but the execution interval of b_j overlaps with all execution intervals of the jobs in $\overline{J_k}$, This gives:

$$\frac{1}{\mu} \sum_{b_i \in \overline{J_k}} A_i(J_k) \geq \frac{1}{\mu}(\mu A_j(J_k)) = A_j(J_k).$$

In any case a blue job gets paid with at least its profit form the green and grey jobs.   □

By the above Lemmata we can now estimate the profit of $J_m(blue)$ in comparison to $J_m(green)$ and $J_m(grey)$. The blue profit is upper bounded by the payment from green and grey jobs, in turn the payment of green and grey jobs is upper bounded by the profits of green and grey jobs.

We have

$$J_m(blue) \leq \left(1 + \frac{2}{\mu}\right)(J_m(green) + J_m(grey)). \tag{5.2}$$

By Lemma 52 we can now express the profit of J_{opt} in terms of $J_m(green)$, only.

$$
\begin{aligned}
|J_{\text{opt}}| &\leq& J_m(blue) + J_m(green) + J_m(grey) \tag{5.3} \\
&\leq& \left(2 + \frac{2}{\mu}\right)(J_m(green) + J_m(grey)) \tag{5.4} \\
&\leq& \frac{2(\mu + 1)}{\mu}\left(J_m(green) + \frac{\mu}{1 - \mu}J_m(green)\right) \tag{5.5} \\
&\leq& \frac{2(\mu + 1)}{\mu}\frac{1}{1 - \mu}J_m(green) \tag{5.6} \\
&\leq& 2\frac{\mu + 1}{\mu(1 - \mu)}J_m(green) \leq 2\frac{\mu + 1}{\mu(1 - \mu)}|J_m|. \tag{5.7}
\end{aligned}
$$

The factor $f(\mu) := 2\frac{\mu+1}{\mu(1-\mu)}$ is minimized for $\mu = \sqrt{2} - 1$ and gives $f(\mu) = 6 + 4\sqrt{2} \approx 11.657$ and our final result.

Theorem 55 *For the geometric firefighter problem inside a simple polygon with non-intersecting barriers there is an approximation algorithms that saves at least $\frac{1}{6+4\sqrt{2}} = \frac{3}{2} - \sqrt{2} \approx 0.086$ times the area of the optimal barrier solution.*

Exercise 21 *Verify the computation of the optimization value $\mu = \sqrt{2} - 1$ and $f(\mu) = 6 + 4\sqrt{2} \approx 11.657$.*

The result is restricted to non-intersecting barriers. Additionally, we require that any point in the polygon that can be covered, can already be covered by a single barrier. Otherwise some barriers $b_1, b_2, \ldots, b_l$ could help each other for covering a part of the polygon that is not covered
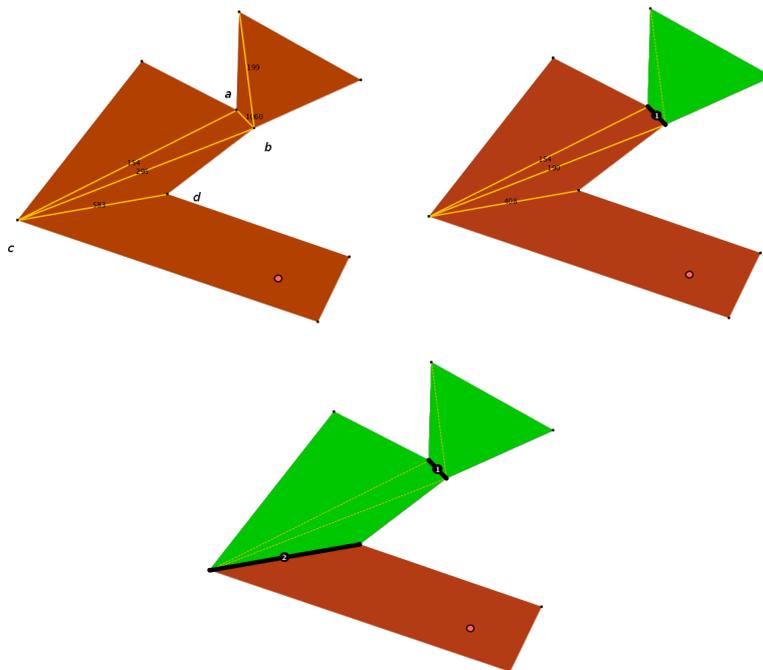
Figure 5.3: After the usage of barrier (a, b), the priority of barrier (c, d) decreases because (a, b) already protects parts of the area related to (c, d).

by a single barrier. In this case we can protect new areas A' which are not related to single barriers. This is obviously not represented in the algorithm and the analysis.

In principle the algorithm could also run for intersecting barriers but there is one major problem. The execution time for building a barrier is not independent from the order of the barrier constructed so far. A barrier might have a release times that depends on the barriers constructed before. For example a barrier b_1 blocks the fire and extends the release time of a barrier b_2 that intersects b_1.

So there might be a barrier of the optimal solution, that contributes to the optimal profit with an arbitrary large amount of its profit but cannot be scheduled in the approximation since it depends on the construction of a very less important (w.r.t. area profit) barrier. The analysis might fail.

Exercise 22 *Construct an example, where the construction of a barrier b_2, depends on the construction of a barrier b_1.*

Exercise 23 *If intersections are allowed, where is the main problem in the proof of the above approximation result?*

Finally, we end the section with an example where the profit of a job is reduced by overlapping, as shown in Figure 5.3.

5.3 Firefighting in the plane

In this section we consider the case that the fire spreads in the plane and we would like to build a firebreak. We assume that there is a single fire that spreads with the same unit speed

speed into any direction – an expanding circle. The configuration was already considered in the Introduction 1.1.3 for circle shaped firebreaks.

Let us assume that we can build arbitrary barrier curves in the plane with speed $v > 1$. The current point has to keep outside the current fire extension. Already constructed parts of the curve serve as obstacles for the expanding fire.

Geometric Firefighter Problem in the plane
Instance: An expanding fire-circle that spreads with unit speed from a given starting point s in the plane and has already reached radius A.
Question: How fast must a firefighter be, to build a firebreak that finally fully encloses and stops the expanding fire?

We assume that we build a single connected curve with a single firefighter.

5.4   On finding an upper bound for speed v

First we discuss some structural properties. If we start a the boundary of the fire and move with speed v, the curve that keeps as close as possible to the fire is a logarithmic spiral with excentricity α for $v = \frac{1}{\cos \alpha}$.

This can be easily seen by the following argument. For a fireman close to the fire, locally not all movements are safe. If the fireman performs an infinitesimal small step x at some point p very close to the fire, the fire grows by distance $\frac{x}{v} = x \cos \alpha$ for $v = \frac{1}{\cos \alpha}$ from the source; see Figure 5.4. This means that with respect to the ray R_q emanating from the source B and running through q the fire fighters can leave R_q *locally* from q only in a direction smaller than or equal to α. If the firefigther continuously exactly move into the direction α, he will follow a logarithmic spiral with excentricity α around B.

In polar-coordinates a logarithmic spiral S is defined by $S(\varphi) := (\varphi, a \cdot e^{\varphi \cot \alpha})$ where a is a constant defined by the distance of the spiral to its origin for the angle $\varphi = 0$. The spiral has the property that the tangent to the spiral at some point $p \in S$ and the ray emanating from the origin and running through p build an angle α for all points $p \in S$. Furthermore, for an origin B and two points q and p on S the path, S_q^p, along S has length

$$|S_q^p| = \frac{1}{\cos \alpha} \left(|Bq| - |Bp| \right) \ . \tag{5.8}$$

This means that a logarithmic spiral of excentricity α that starts at some point q close to the fire circle of radius $|Bq|$ is always a correct firebreak. Additionally, the logarithmic spiral with excentricity $\alpha \in (0, \pi/2]$ that starts at some point q will always finally hit a line l' running through q; see Figure 5.4.

Exercise 24 *Let us assume that a natural firebreak is already given by a line and the fire expands in one half-plane. How fast must the firefighter be, in order to enclose the fire in this scenario?*

5.4.1   The limit speed

The following considerations are take from Bressan et al. 2008. The above discussion means that logarithmic spirals might be good candidates for enclosing an expanding circle.

We consider the following situation for a given spiral excentricity $\beta \in (\pi/2, 0)$ (for which $\lambda_\beta := \cot \beta$ runs from 0 to infinity). Let us assume that a spiral firebreak was constructed from the origin up to point D with some speed v and reaches D at some time t. Now the fire starts and also mets the point D after the same time t.
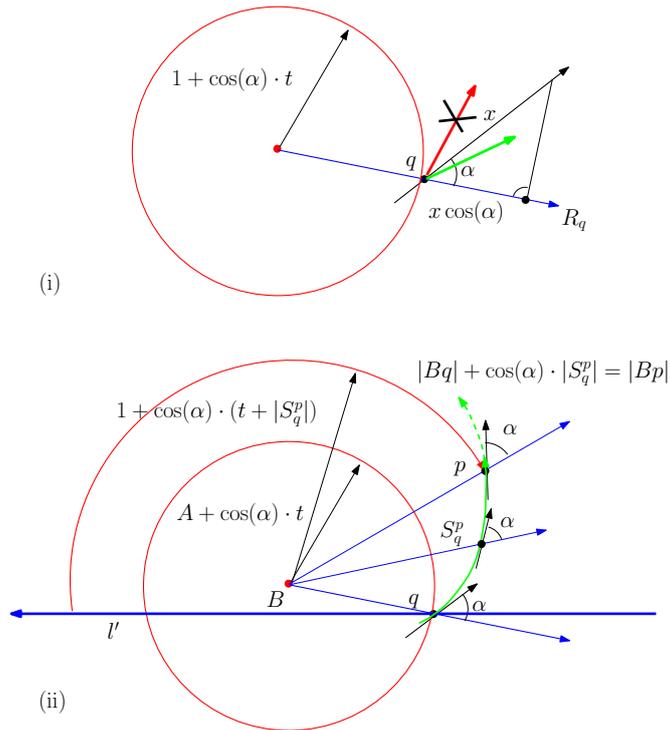
Figure 5.4: (i) At some point q close to the fire, locally the firefigther can leave the ray R_q at q only in a direction smaller than or equal to α. (ii) If the firefigthers continuously exactly move into direction α, they follow a logarithmic spiral with excentricity α around B which results in a correct firebreak.

Intuitively, we assume that the fire starts with some small delay and runs along the already constructed spiral to the point $p = (0, 1)$ and then follows the tangent at p and always reaches the already constructed spiral arc of the next round at some point $D = (2\pi + \gamma, a \cdot e^{(2\pi+\gamma)\cot\beta})$ as depicted in Figure 5.5. For the delay in the very beginning we obviously have to start the fire only one round later, below we would like to construct a correct starting situation. Since we can scale the situation arbitrarily, the overall absolute delay can be arbitrarily small and will be covered by speed. We would like to find a necessary condition for the speed for moving on.

By the law of sine we calculate the corresponding angle $\gamma := \gamma(\beta) \in (0, \pi/2)$ implicity by

$$\frac{a \cdot e^{(2\pi+\gamma)\cot\beta}}{\sin\beta} = \frac{a}{\sin(\beta-\gamma)} \iff e^{(2\pi+\gamma)\cot\beta} = \frac{\sin\beta}{\sin(\beta-\gamma)} \, . \tag{5.9}$$

By construction the angle γ exists for any spiral and it grows continuously in the interval $(0, \pi/2)$ for β in $(0, \pi/2)$. We would like to compute, how fast we have to move along the spiral so that the fire following us along the spiral will always exactly met us at point D. (Speed less than this value will let the fire overrun us at point D.)

This means, that we consider the ratio

$$f(\beta) := \frac{|\text{Length of the spiral from } O \text{ to } D|}{|\text{Length of the spiral from } O \text{ to } C| + |\text{Length of the segment } CD|} \, . \tag{5.10}$$

We express CD by $a\frac{\sin\gamma}{\sin\beta}e^{(2\pi+\gamma)\cot\beta}$ by the law of sine and Equation 5.10 reads as follows:

$$f(\beta) = \frac{\frac{1}{\cos\beta}e^{(2\pi+\gamma)\cot\beta}}{\frac{1}{\cos\beta} + \frac{\sin\gamma}{\sin\beta}e^{(2\pi+\gamma)\cot\beta}} = \frac{\frac{1}{\cos\beta}\frac{\sin\beta}{\sin(\beta-\gamma)}}{\frac{1}{\cos\beta} + \frac{\sin\gamma}{\sin\beta}\frac{\sin\beta}{\sin(\beta-\gamma)}} = \frac{1}{\cos\gamma} \, . \tag{5.11}$$
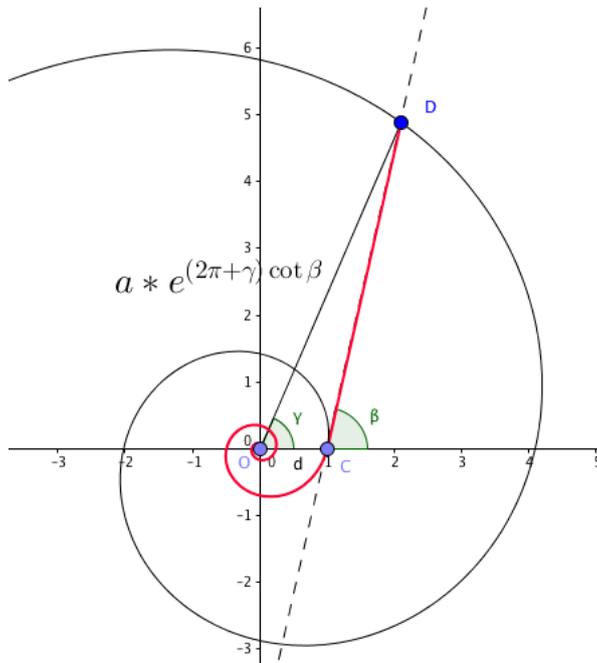
Figure 5.5: The fire starts with a delay of one round and follows the boundary of the spiral. How fast do we have to move so that the fire cannot overrun us at point D? Let $a = \overline{OC}$. By the definition of the spiral and the law of sine we can compute the corresponding lenghts.

Let us consider the behaviour of $f(\beta)$ at the boundary. If β goes to 0 and $\cot \beta$ goes to infinity, we argue that $\gamma(\beta)$ also goes to zero, which means

$$\lim_{\beta \mapsto 0} f(\beta) = \lim_{\gamma \mapsto 0} \frac{1}{\cos \gamma} = 1 \,. \tag{5.12}$$

If β goes to $\pi/2$ from below, $\cot \beta$ goes to 0 but remains positive, $\gamma(\beta)$ is smaller than $\pi/2$. Thus,

$$\limsup_{\beta \mapsto \pi/2 \nearrow} f(\beta) \le \limsup_{\beta \mapsto \pi/2 \nearrow} e^{(2\pi+\gamma)\cot \beta} \le \lim_{\beta \mapsto \pi/2 \nearrow} e^{(5\pi/2)\cot \beta} = 1 \,. \tag{5.13}$$

This means that f is a continuous function in $\beta = (0, \pi/2)$ and there will be some unique global maximum

$$v_l := \max_{\beta \in (0,\pi/2)} f(\beta) \,.$$

This global maximum is found numerically by $\beta_l = 1.29783410242\ldots$ and gives $v_l = f(\beta_l) = 2.614430844\ldots$ and $\gamma(\beta_l) = 1.178303978\ldots$ A sketch of the plot of $f(\beta)$ is shown in Figure 5.6.

This means that in the above scenario, we can assume that if we construct a spiral of some arbitrary excentricity β with speed $v > v_l$ for t time steps and after that let the fire start at the origin and spread also time t, the fire will keep really behind the current point F as depicted in Figure 5.7.

Exercise 25 *Let us assume that in the firefighter problem the fire and the firefighter start at the same point and the fire starts with some delay, so that we can build a spiral. Then the fire starts and make up time, so that finally the fire moves with unit speed.*
Is the spiral with excentricity β and speed $f(\beta)$ always a legal strategy? Are there other reasonable strategies in this case?
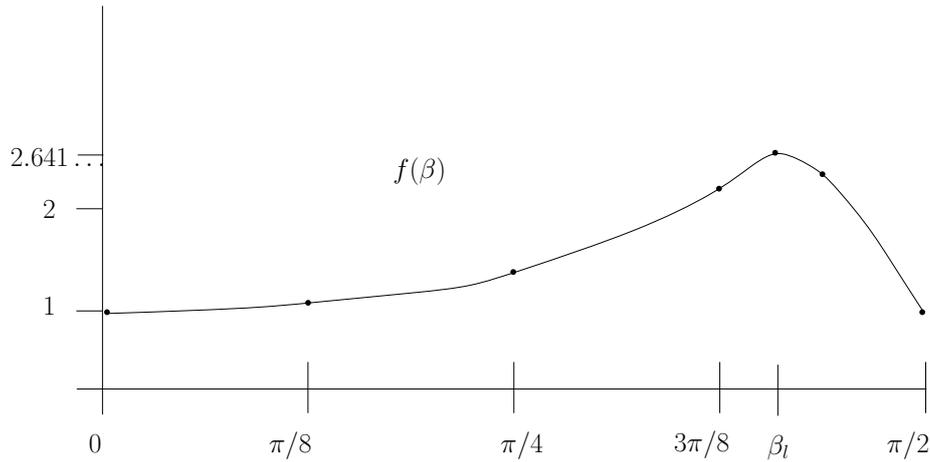
Figure 5.6: A sketch of the plot of the curve $f(\beta)$ for $\beta \in (0, \pi/2)$. There is a unique maximum $2.614430844\ldots$ for $\beta_l = 1.29783410242\ldots$

Now we would like to argue that for speed $v > v_l$ we always also find a correct starting situation that resembles the firefighting problem.

1. The starting situation

Let us assume that we have speed $v_1 = \frac{1}{\cos \beta_1} > v_l$ and we use a spiral with excentricity β_1. Then there is always a speed $v' = \frac{1}{\cos \beta'}$ with $v_1 > v' > v_l$ which allows us to use the spiral for β_1 as follows. If the firefighter moves with speed $v' < v_1$ for the common start from the origin the fire will not reach the current point F

Consider Figure 5.8. Let us assume that for some time t' and speed $f(\beta_1)$ both, firefighter and fire met point F by starting at the origin. Thus, for speed $v' = \frac{1}{\cos \beta'}$ and meeting F the fire would have reached some point point D and with speed $\frac{1}{\cos \beta_1}$ some point N and the time difference x reads $t(\cos \beta' - \cos \beta_1)$ for arbitrary large t. More precisely, F is met at time $t_1 = t \cos \beta_1$ and time $t_2 = t \cos \beta' > t_1$, respectively and $t_2 - t_1$ gives the time difference for the fire, the time used for moving from N to D.

We use this time difference to construct the starting situation, so that the fire and the firefighter do not start at the same point. Thus we attain a correct starting situation that resembles the firefighting problem.

There are always small values ϵ_1 and ϵ_2 so that

$$\frac{1}{\cos \beta'}(\epsilon_1 + \epsilon_2) - \epsilon_2 < x$$

holds. We allow the fire to start a bit earlier while we start away from the fire with speed $\frac{1}{\cos \beta_1}$. The fire should only reach D when the firefighter is located at F.

The starting situation is given in Figure 5.9. The fire has starting radius ϵ_1. We start with the spiral of excentricity β_1 at distance $\epsilon_1 + \epsilon_2$ away from the fire at point some point B. The firefighter starts at point B and the fire starts spreading at the same time. The fire will reach B after ϵ_2 time steps, and the firefighter have already moved a distance of $\frac{\epsilon_2}{\cos \beta_1}$. Additionally, the excentricity β_1 takes care, that we do not move into the fire by accident.

For speed $\frac{1}{\cos \beta'}$, the time saved by the fire is $\frac{1}{\cos \beta'}(\epsilon_1 + \epsilon_2) - \epsilon_2$. But this helps the fire only to visit point D, if we use speed $\frac{1}{\cos \beta_1}$.
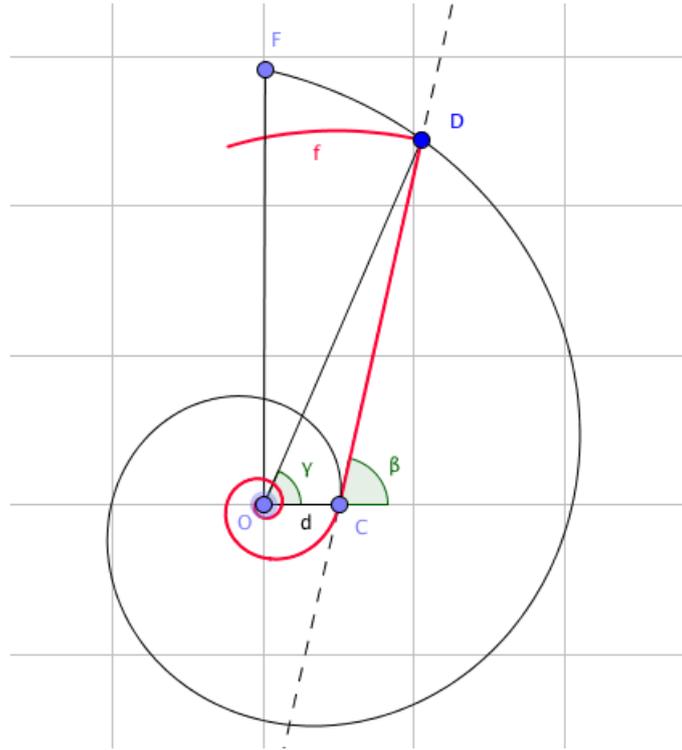
Figure 5.7: For speed $v > v_l$ the fire keeps behind the current point F.
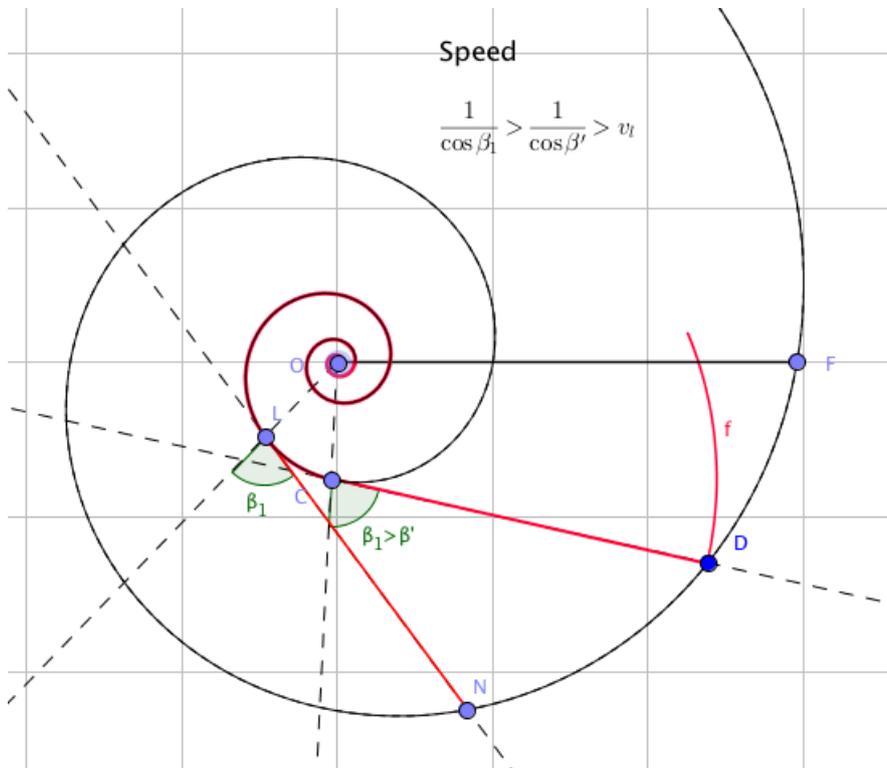
Figure 5.8: We can start the spiral with a slightly smaller speed. The current point is still not met and a time difference x can be used for the starting situation.
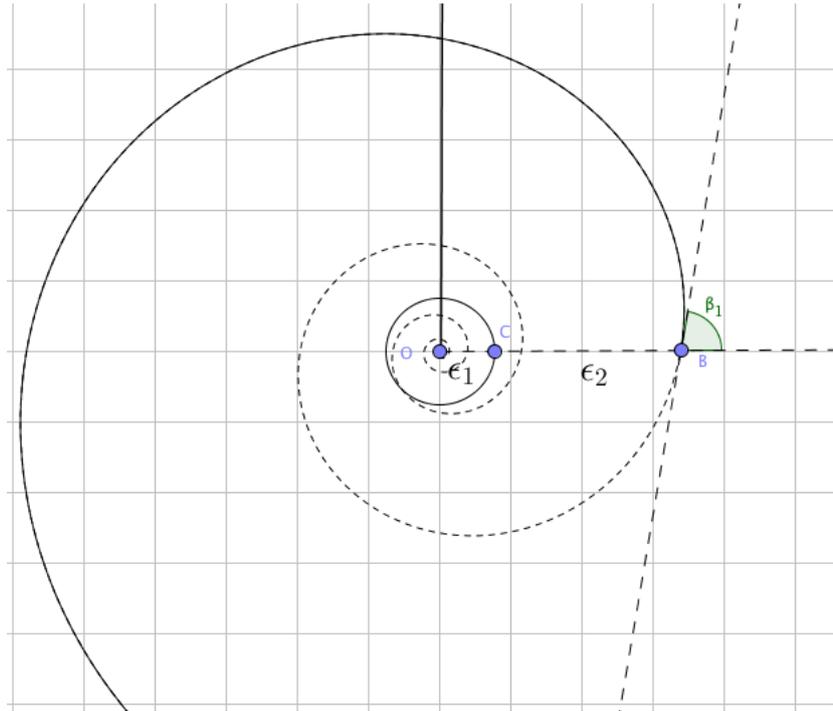
Figure 5.9: We make use of the time difference x and attain an admissable starting situation as given in Figure 5.10.

Altogether, we can assume that for speed $v_1 = \frac{1}{\cos \beta_1}$ we can start a spiral strategy outside the fire circle, and we reach some point $(e^{2k_1 \pi \cot \beta_1}, 0)$ whereas the fire spreads up to some point D behind the current location F of the fire fighter; see Figure 5.10.

Exercise 26 *Show that there is also an admissable starting situation, where the firefighter starts at the boundary of a fire-circle with radius $> \epsilon$.*

2. An iteration process for a successful strategy

Here we sketch the proof given by Bressan et al. 2008. The first spiral starts with radius $C_1 := \epsilon_1 + \epsilon_2$ and excentricity β_1 and ends at some point $F = (C_1 e^{2k_1 \pi \cot \beta_1}, 0)$. The fire keeps behind at some point D as shown in Figure 5.11.

We now start another spiral at $F = (e^{2k_1 \pi \cot \beta_1}, 0)$ with excentricity $\beta_2 > \beta_1$ and starting radius $C_2 = C_1 e^{2k_1 \pi \cot \beta_1}$. This means that we make use of a slightly tigther logarithmic spiral. If β_2 is small enough, this spiral strategy will also be a correct intermediate strategy.

We let the spiral with excentricity β_2 run up to some angle $2k_2\pi$, thus attaining the same situation again. The strategy ends at some point F_2 and the fire is behind at some point D_2; see Figure 5.12.

After scaling, we assume that we have the same situation again. This means that we apply the same arguments and attain a sequence of increasing angles $\beta_1 < \beta_2 < \ldots < \beta_l$ and successive spirals $C_i e^{2\pi k_i \cot \beta_i}$ that result in a correct strategy.

Finally, we can also scale so that the overall strategy ends at point $(e^{2\pi \cot \beta_m}, 0)$. We would like to argue that for some β_m large enough there is time enough to move from $(e^{2\pi \cot \beta_m}, 0)$ to the point $(1, 0)$ on the previous round. This means that the fire did not arrive at $(1, 0)$; see Figure 5.13.
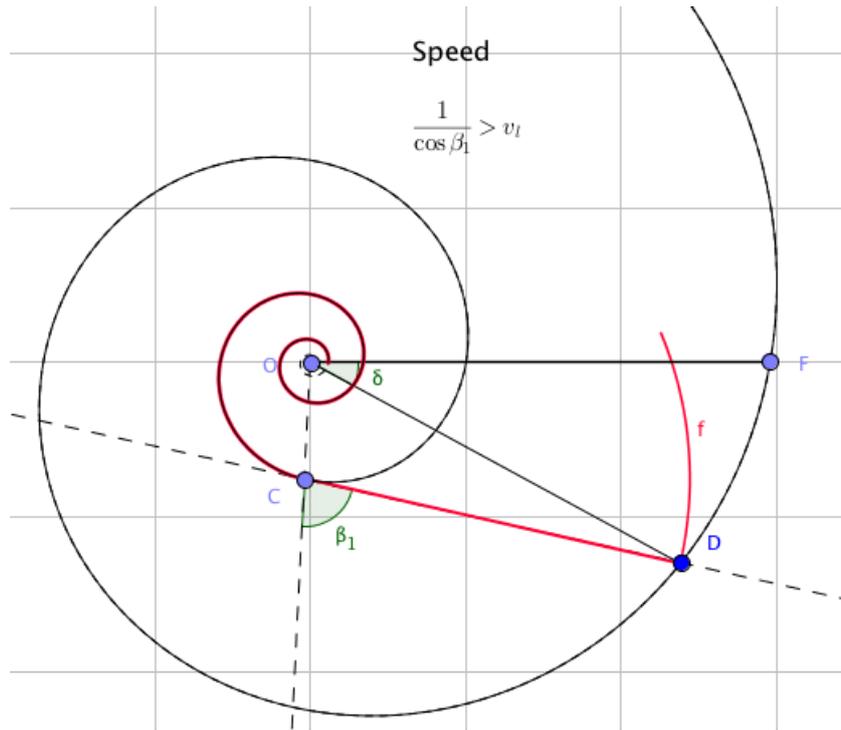
Figure 5.10: An admissable starting situation for speed $v_1 = \frac{1}{\cos \beta_1}$.
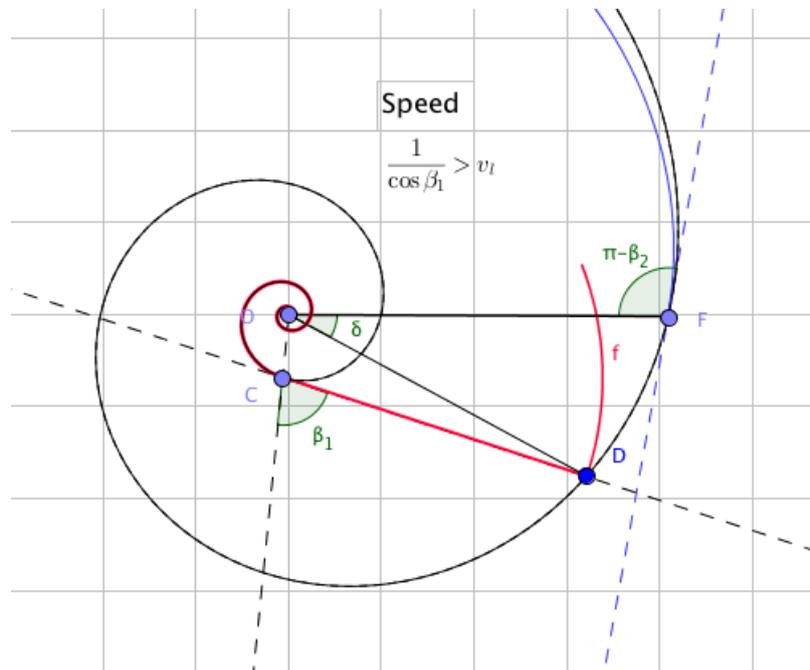
Figure 5.11: We start with a second spiral and excentricity $\beta_2 > \beta_1$.
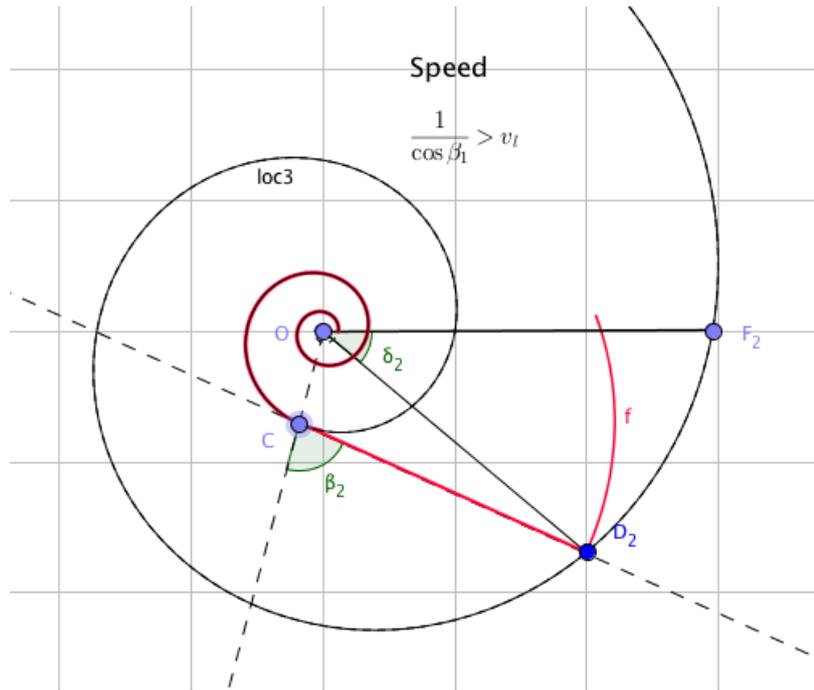
Figure 5.12: By scaling we attain the same situation again.
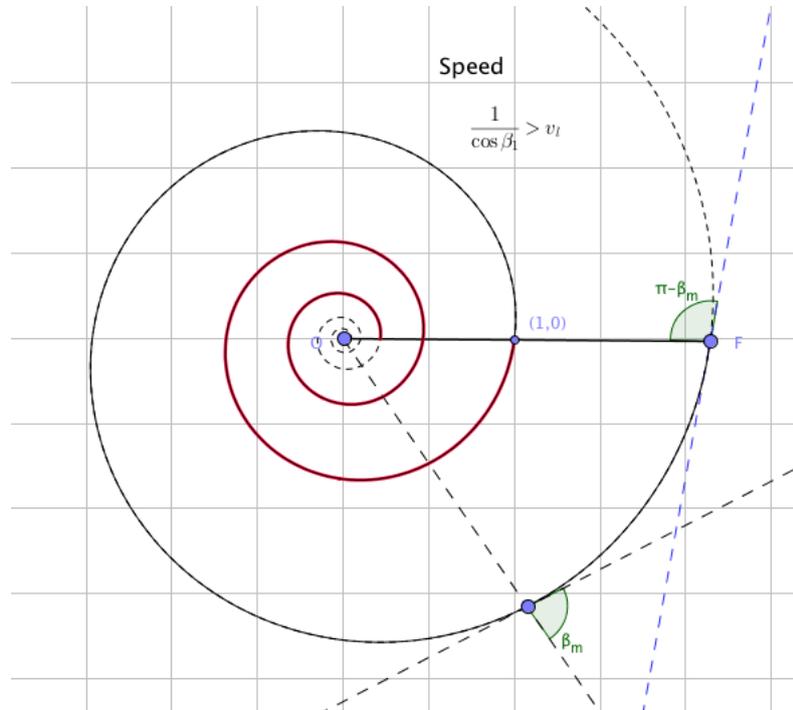
Figure 5.13: The firefighter ends with a spiral of excentricity $\beta_m > \beta_{m-1} > \cdots > \beta_1$. The angle β_m is large enough so that the fire on the previous round did not arrive at $(1,0)$ when the firefighter moves from F to $(1,0)$ along the line segment.
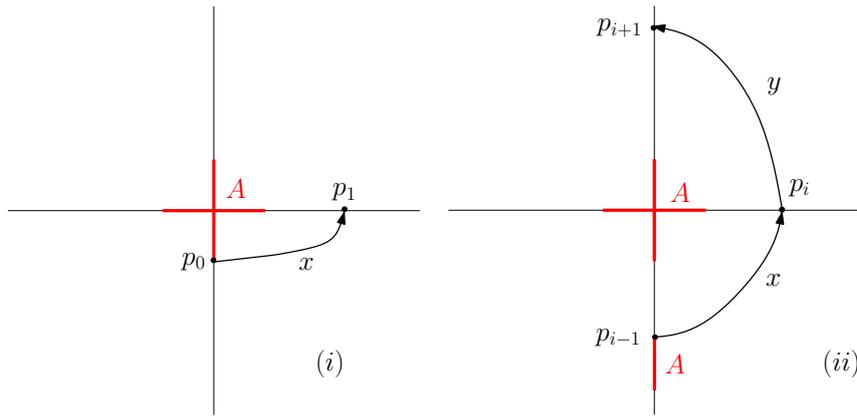
Figure 5.14: Proof of Theorem 57.

The running time of the overall last spiral is smaller than $\frac{1}{\cos \beta_m} e^{2\pi \cot \beta_m}$. Additionally, moving to point $(1,0)$ yields $e^{2\pi \cot \beta_m} - 1$. For reaching point $(1,0)$ the fire also has to a distance arbitrarily close to $\frac{1}{\cos \beta_m}$ for following the last spiral, this follows from scaling. If β_m grows, there will be some β_m that finally fulfills

$$\frac{1}{\cos \beta_m} > \frac{1}{\cos \beta_1} \left(\frac{1}{\cos \beta_m} e^{2\pi \cot \beta_m} + (e^{2\pi \cot \beta_m} - 1) \right) \tag{5.14}$$

which gives the conclusion. For example for $\beta_1 \approx 1.191388\ldots$ and $\frac{1}{\cos \beta_1} = 2.7$ we require $\beta_m > 1.4268$.

Theorem 56 *(Bressan et al. 2008) For any speed $v > v_l \approx 2.614430844$ there is a spiralling strategy that finally encloses an expanding circle that expands with unit speed.*

5.5   A simple lower bound for spiralling strategies

The following results stem from Klein et al. 2015. A barrier building strategy S is called *spiralling* if it starts on the boundary of a fire of radius A, and visits the four coordinate half-axes in counterclockwise order and at increasing distances from the origin.

Now let S be any spiralling strategy of maximum speed $v \leq (1 + \sqrt{5})/2 \approx 1.618$, the golden ratio. We can assume that S proceeds at constant speed v. Let $p_0, p_1, p_2, \ldots$ denote the points on the coordinate axes visited, in this order, by S. The following Lemma shows that S cannot succeed because there is still fire burning outside the barrier on the axis previously visited.

Lemma 57 *Let A be the initial fire radius. When S visits point p_{i+1}, the interval $[p_i, p_i + \mathtt{sign}(p_i)A]$ on the axis visited before is on fire.*

Proof. The proof is by induction on i. Suppose strategy S builds a barrier of length x between p_0 and p_1, as shown in Figure 5.14 (i). During this time the fire advances x/v along the positive X-axis, so that $A + x/v \leq p_1 \leq x$ must hold, or

$$\frac{x}{v} \geq \frac{1}{v-1} A > A;$$

the last inequality follows from $v < 2$. Thus, the fire has enough time to move a distance of A from p_0 downwards along the negative Y-axis.
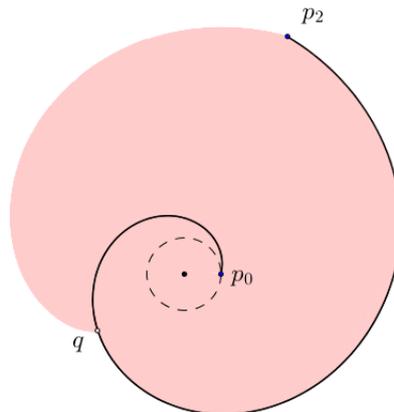
Figure 5.15: The race between the fire and the fighter. When the fighter arrives at point p_2, having constructed a firebreak from p_0 to p_2, the fire has expanded along the outer side of the barrier up to point q. Will the fire fighter be able to contain the fire?

Now let us assume that strategy S builds a barrier of length y between p_i and p_{i+1}, as shown in Figure 5.14 (ii). By induction, the interval of length A below p_{i-1} is on fire. Also, when the fighter moves on from p_i, there must be a burning interval of length at least $A + x/v$ on the positive Y-axis which is not bounded by a barrier from above. This is clear if p_{i+1} is the first point visited on the positive Y-axis, and it follows by induction, otherwise. Thus, we must have $A + x/v + y/v \leq p_{i+1} \leq y$, hence

$$\frac{y}{v} \geq \frac{1}{v-1} A + \frac{1}{v(v-1)} x > A + x.$$

The rightmost inequality follows since v is supposed to be smaller than the golden ratio, which satisfies $X^2 - X - 1 = 0$; hence, $v^2 - v < 1$. This shows that the fire has time to crawl along the barrier from p_{i-1} to p_i, and a distance A to the right, as the fighter moves to p_{i+1}, completing the proof of Theorem 58. $\square$

Theorem 58 *In order to enclose the fire, a spiralling strategy must be of speed*

$$v > \frac{1+\sqrt{5}}{2} \approx 1.618,$$

the golden ratio.

5.6 An alternative approach: The firefighter curve

In this section we describe an alternative approach and a special strategy for the upper bound $v \approx 2.614430844$. The result also stems from Klein et al. 2015. We construct a barrier curve that always keeps as close as possible to the fire. In the beginning this gives a logarithmic spiral of excentricity α for speed $v = \frac{1}{\cos \alpha}$.

In our case while the fighter keeps building the barrier, the fire is coming after her along the outside of the barrier, as shown in Figure 5.15. Intuitively, the fighter can only win this race, and contain the fire, if the last coil of the barrier hits the previous coil. In the example in Figure 5.16, this happens in the second round if $v = 4.1932$; but for smaller values of v, more rounds may be necessary. In this section we prove the following result.
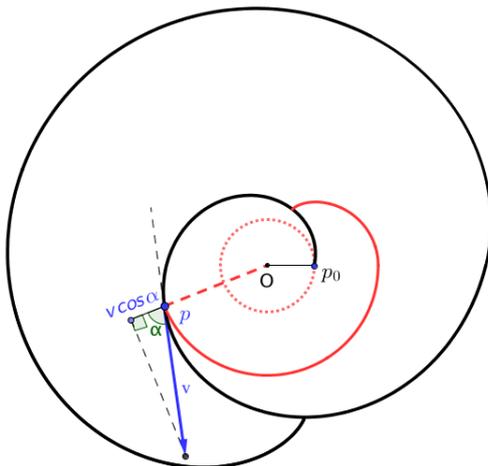
Figure 5.16: At speed $v = 4.1932$ the fire will be fully contained by the fire fighter's barrier in the second round. Angle α is constant because $v \cos \alpha$ equals 1, the fire's expansion speed, by definition of strategy FF.

Theorem 59 *Strategy FF contains the fire if $v > v_c \approx 2.6144$ holds.*

5.6.1   The first rounds

Let p be a point on the barrier curve's first round, as depicted in Figure 5.16, (ii). If α denotes the angle between the fighter's velocity vector at p and the ray from 0 through p, the fighter moves at speed $v \cos \alpha$ away from 0. This implies $v \cos \alpha = 1$, because the fire expands at unit speed and the fighter stays on its frontier, by definition of strategy FF. Since the fighter is operating at constant speed v, angle α is constant, and given by $\alpha = \cos^{-1}(1/v)$.

Consequently, the first part of the barrier curve, between points p_0 and p_1 shown in Figure 5.17, (i), is part of a logarithmic spiral of excentricity α centered at 0. In polar coordinates, this segment can be desribed by $(\varphi, A \cdot e^{\varphi \cot \alpha})$, where $\varphi \in [0, 2\pi]$, and A denotes the distance from the origin to p_0, i.e., the fire's intitial radius.

In general, the curve length of a logarithmic spiral of excentricity α between two points at distance $d_1 < d_2$ to its center is known to be $\frac{1}{\cos \alpha}(d_2 - d_1)$. Thus, we have for the length l_1 of the barrier curve from p_0 to p_1 the equation

$$l_1 \;=\; \frac{A}{\cos(\alpha)} \cdot (e^{2\pi \cot(\alpha)} - 1) \tag{5.15}$$

From point p_1 on, the geodesic shortest path, along which the fire spreads from 0 to the fighter's current position, p, is no longer straight. It starts with segment $0p_0$, followed by segment p_0p, until, for $p = p_2$, segment p_0p becomes tangent to the barrier curve at p_0; see Figure 5.17, (ii). By the same arguments as above, between p_1 and p_2 the barrier curve constructed by FF is also part of a logarithmic spiral of excentricity α, but now centered at p_0. This spiral segment starts at p_1 at distance $A' = A(e^{2\pi \cot \alpha} - 1)$ from its center p_0. Since p_2 and p_1 form an angle α at p_0, the distance from p_2 to p_0 equals $A'e^{\alpha \cot \alpha}$. Thus, the curce length from p_1 to p_2 is given by $l'_2 = \frac{A'}{\cos \alpha}(e^{\alpha \cot \alpha} - 1) = \frac{A}{\cos \alpha}(e^{2\pi \cot \alpha} - 1)(e^{\alpha \cot \alpha} - 1)$. Consequently, the overall curve length l_2 from p_0 to p_2 equals

$$l_2 \;=\; l_1 + l'_2 \;=\; \frac{A}{\cos \alpha}(e^{2\pi \cot \alpha} - 1)e^{\alpha \cot \alpha} \tag{5.16}$$
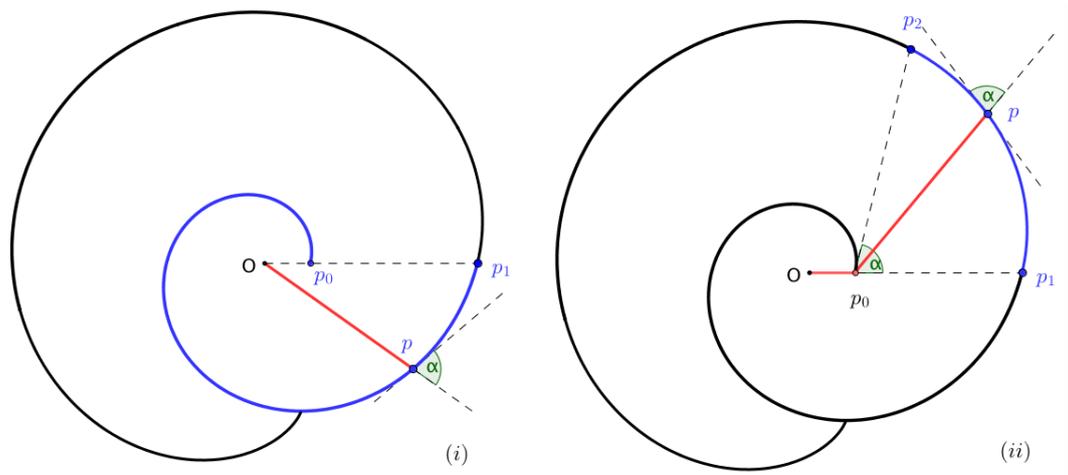
Figure 5.17: The barrier curve starts with two parts of logarithmic spirals of excentricity α, centered at 0 and p_0, respectively.
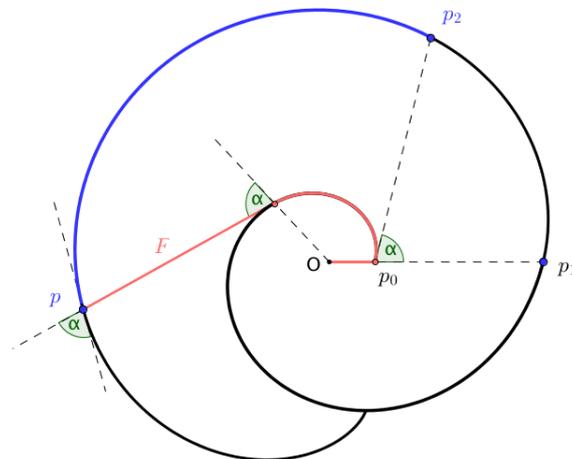
Figure 5.18: From point p_2 on the barrier curve results from a wrapping around the already constructed barrier. The last segment, *free string F*, of the shortest path from the fire source to the current barrier point p shrinks, by wrapping, and simultaneously grows by $\cos \alpha$. The barrier curve starting from p_2 is no longer a logarithmic spiral. The strategy will be successful if F shrinks to zero.

Figure 5.19: Repeatedly constructing backwards tangents may end in 0 or in p_0. This way, two types of linkages are defined.

From point p_2 on, the geodesic shortest path from 0 to the fighter's current position, p, starts wrapping around the existing spiral part of the curve, beginning at p_0; see Figure 5.18. The last segment of this path is tangent to the previous round of the curve. As mentioned in the Introduction, we shall endeavor to determine its length, F, because the fire will be contained if and only if F ever attains the value 0.

One could think of this tangent as a string (named the *free string*) at whose endpoint, p, a pencil is attached that draws the barrier curve. But unlike an involute, here the string is not normal to the outer layer. Rather, its extension beyond p forms an angle α with the barrier's tangent at p. This causes the string to grow in length by $\cos \alpha$ for each unit drawn. At the same time, the inner part of the string gets wrapped around the previous coil of the barrier. It is this interplay between growing and wrapping that we will investigate below.

One can show that after p_2 the barrier curve is no longer segment of a logarithmic spiral. But to give a closed form representation for the second round, leave alone for subsequent rounds recursively constructed, seems to be out of reach.

At the end of this subsection, let us recall the following fact. As the fighter is building the barrier at speed $v = 1/\cos \alpha$, the fire is coming after her at unit speed along the outside of the barrier, as indicated in Figure 5.15. Thus, each barrier point p is caught by fire twice, once from the inside, when the fighter passes through p, and a second time from the outside, if the fire is not stopped before.

5.6.2   Structural properties

In this subsection we assume that the fighter has built quite a few rounds of the barrier curve without yet containing the fire. That the first two rounds of the curve involve two different spiral segments, around 0 and around p_0, carries over to subsequent layers. The structure of the curve can be described as follows. Let l_1 and l_2 denote the curve lengths from p_0 to p_1 and p_2, respectively, as in Equations 5.15 and 5.16. For $l \in [0, l_1]$ let $F_0(l)$ denote the segment connecting 0 to the point of curve length l; see the sketch given in Figure 5.19. At the endpoint

of $F_0(l)$ we construct the tangent and extend it until it hits the next layer of the curve, creating a segment $F_1(l)$, and so on. This construction gives rise to a "linkage" connecting adjacent layers of the curve. Each edge of the linkage is turned counterclockwise by α with respect to its predecessor. The outermost edge of a linkage is the free string mentioned above. As parameter l increases from 0 to l_1, edge $F_0(l)$, and the whole linkage, rotate counterclockwise. While $F_0(0)$ equals the line segment from the origin to p_0, edge $F_0(l_1)$ equals segment $0p_1$.

Analogously, let $l' \in [l_1, l_2]$, and let $\phi_0(l')$ denote the segment from p_0 to the point at curve length l' from p_1. This segment can be extended into a linkage in the same way. We observe that

$$F_{j+1}(l_1) = \phi_{j+1}(l_1) \tag{5.17}$$
$$F_{j+1}(0) = \phi_j(l_2) \tag{5.18}$$

hold (but initially, we have $F_0(l) = A + \cos(\alpha)\, l$ and $\phi_0(l') = \cos(\alpha)\, l'$, so that $F_0(l_1) \neq \phi_0(l_1)$). Clearly, each point on the curve can be reached by a unique linkage, as tangents can be constructed backwards. We refer to the two types of linkages by F-type and ϕ-type. As Figure 5.19 illustrates, points of the same linkage type form alternating intervals along the barrier curve. If p's linkage is of F-type then p is uniquely determined by the index $j \geq 0$ and parameter $l \in [0, l_1]$ such that p is the outer endpoint of edge $F_j(l)$.

Now we will derive two structural properties of F-linkages on which our analysis will be based; analogous facts hold for ϕ-linkages, too. To this end, let $L_j(l)$ denote the length of the barrier curve from p_0 to the outer endpoint of edge $F_j(l)$, and let $F_j(l)$ also denote its length of edge $F_j(l)$.

Lemma 60 *We have $L_{j-1}(l) + F_j(l) = \cos\alpha\, L_j(l)$.*

Proof. Both, fire and fire fighter, reach the endpoint of $F_j(l)$ at the same time. The fire has travelled a geodesic distance of $L_{j-1}(l) + F_j(l)$ at unit speed, the fighter a distance of $L_j(l)$ at speed $1/\cos\alpha$. $\qquad\square$

Equivalently, one could argue that the free string grows by $\cos\alpha$ for each barrier unit built; thus, the total string length (wrapped plus free) of $L_{j-1}(l) + F_j(l)$ must be equal to $\cos\alpha\, L_j(l)$.

The second property is related to the wrapping of the free string. Intuitively, it says that if we turn an F-linkage, the speed of each edge's endpoint is proportional to its length (with the same proportionality constant for all j depending on turning speed).

Lemma 61 *As functions in l, L_j and F_j satisfy the following equation.*

$$\frac{L'_{j-1}(l)}{L'_j(l)} = \frac{F_{j-1}(l)}{F_j(l)}.$$

First, we will derive Lemma 61 from two general facts on smooth curves stated in Lemma 62 and Lemma 63.

Lemma 62 *Suppose a string of length F is tangent to a point t on some smooth curve C. Now the end of the string moves a distance of ϵ in the direction of α, as shown in Figure 5.20. Then for the curve length $C_t^{t_\epsilon}$ between t and the new tangent point, t_ϵ, we have*

$$\lim_{\epsilon \to 0} \frac{C_t^{t_\epsilon}}{\epsilon} = \frac{\sin\alpha\, r}{F}$$

where r denotes the radius of the osculating circle at t.

Figure 5.20: A string wrapping around a curve.

Figure 5.21: Intersection of turned normals.

This fact is quite intuitive. The more perpendicular the motion of the string's endpoint, and the larger the radius of curvature, the more string gets wrapped. But if the string is very long, the effect of the motion decreases; see also Figure 5.20.

The center of the osculating circle at t is known to be the limit of the intersections of the normals of all points near t with the normal at t. Lemma 63 shows what happens if, instead of the normals, we consider the lines turned by the angle $\pi/2 - \alpha$.

Lemma 63 *Let t be a point on a smooth curve C, whose osculating circle at t is of radius r. Consider the lines L_s resulting from turning the normal at points s by an angle of $\pi/2 - \alpha$. Then their limit intersection point with L_t has distance $\sin \alpha\, r$ to t.*

A simple example is shown in Figure 5.21 for the case where curve C itself is a circle. Now we can prove Lemma 61.

Proof.[Proof of Lemma 61] By Lemma 62, applied to the inner point t of edge $F_j(L_j)$, we have

$$\frac{L'_{j-1}(L_j)}{L'_j(L_j)} \;=\; L'_{j-1}(L_j) \;=\; \frac{\sin \alpha\; r}{F_j(L_j)}.$$

Lemma 63 implies that $\sin\alpha\ r$ equals the distance between t and the limit intersection point of the normals turned by $\pi/2 - \alpha$ near t. But for the barrier curve generated by strategy FF, these turned normals are the tangents to the previous coil, so that $\sin\alpha\ r = F_{j-1}(L_j)$ holds. As we substitute variable L_j with $L_j(l)$, the derivatives of the inner functions cancel out and we obtain Lemma 61. $\qquad\square$

The proofs of Lemma 63 and 62 are rather straightforward. We include them for completeness.

Proof.[Proof of Lemma 62] Using the notations in Figure 5.20, the following hold. From $r\sin(\phi/2) = s = a\cos(\phi/2)$ we obtain $a = r\tan(\phi/2)$. For short, let $c := C_t^{t_\epsilon}$. By l'Hospital's rule,

$$\frac{c}{2a} = \frac{r\,\phi}{2r\,\tan(\phi/2)} \approx \cos^2(\phi/2) \to 1$$

as ϵ, hence ϕ, go to 0. Thus, $2a$ is a good approximation of $c = C_t^{t_\epsilon}$. By the law of sines,

$$\frac{\epsilon\sin(\alpha)}{\sin(\phi)} = \frac{F_\epsilon + a}{\sin(\pi/2)},$$

hence

$$\frac{\sin(\phi)}{\epsilon} = \frac{\sin(\alpha)}{F_\epsilon + a} \to \frac{\sin(\alpha)}{F}$$

This implies $\sin(\phi/2)/\epsilon \to \sin(\alpha)/(2F)$, and we conclude

$$\frac{C_t^{t_\epsilon}}{\epsilon} = \frac{c}{2a}\frac{2a}{\epsilon} \approx \frac{2r\tan(\phi/2)}{\epsilon} = \frac{2r\sin(\phi/2)}{\epsilon\cos(\phi/2)} \to \frac{r\sin(\alpha)}{F}.$$

$\qquad\square$

Proof.[Proof of Lemma 63] Let us assume that C is locally parameterized by $Y = f(X)$ and that $t = (x_0, f(x_0))$. Then the tangent in t is

$$Y = f'(x_0)X - f'(x_0)x_0 + f(x_0),$$

and line L_t, the tangent turned counterclockwise by α, is given by

$$Y = \tan(\arctan(f'(x_0)) + \alpha)X - \tan(\arctan(f'(x_0)) + \alpha)x_0 + f(x_0).$$

Now let (v, w) denote the point of intersection of L_t and L_s, where $s = (x_0 + \epsilon, f(x_0 + \epsilon))$. Equating the two line equations we obtain

$$\big(h(x_0 + \epsilon) - h(x_0)\big)\,v = g(x_0 + \epsilon) - g(x_0) + f(x_0) - f(x_0 + \epsilon)$$

where

$$h(x) := \tan(\arctan(f'(x)) + \alpha) \quad\text{and}\quad g(x) := h(x)x$$

After dividing by ϵ and taking limits, we have

$$h'(x_0)\,v_0 = g'(x_0) - f'(x_0) = h'(x_0)\,x_0 + h(x_0) - f'(x_0),$$

which results in

$$
\begin{aligned}
v_0 &= x_0 + \frac{h(x_0) - f'(x_0)}{h'(x_0)} \\
w_0 &= h(x_0)\,v_0 - g(x_0) + f(x_0) \\
&= f(x_0) + \frac{h^2(x_0) - h(x_0)f'(x_0)}{h'(x_0)}.
\end{aligned}
$$

In other words,

$$(v_0, w_0) - (x_0, f(x_0)) \;\;=\;\; \frac{h(x_0) - f'(x_0)}{h'(x_0)} (1, h(x_0))$$

$$|(v_0, w_0) - (x_0, f(x_0))| \;\;=\;\; |\frac{h(x_0) - f'(x_0)}{h'(x_0)}| \sqrt{1 + h^2(x_0)}$$

Using the addition formula for tan,

$$h(x) \;\;=\;\; \tan(\arctan(f'(x)) + \alpha) \;\;=\;\; \frac{f'(x) + \tan(\alpha)}{1 - f'(x)\tan(\alpha)},$$

we obtain

$$h(x_0) - f'(x_0) \;\;=\;\; \frac{1 + (f'(x_0))^2 + \tan(\alpha)}{1 - f'(x_0)\tan(\alpha)}.$$

and

$$1 + h^2(x_0) \;\;=\;\; \frac{(1 + (f'(x_0))^2)(1 + \tan^2(\alpha))}{(1 - f'(x_0)\tan(\alpha))^2}.$$

Moreover,

$$h'(x_0) \;\;=\;\; \frac{f''(x)(1 + \tan^2(\alpha))}{(1 - f'(x_0)\tan(\alpha))^2}.$$

Putting expressions together we obtain

$$|(v_0, w_0) - (x_0, f(x_0))| \;\;=\;\; |\frac{\left(1 + (f'(x_0))^2\right)^{3/2}}{f''(x_0)}| \frac{\tan(\alpha)}{\sqrt{1 + \tan^2(\alpha)}}.$$

The first term is known to be the radius of the osculating circle, r, and the second equals $\sin(\alpha)$.

$\square$

5.6.3   Differential equations

In this section we turn the structural properties observed in Subsection 5.6.2 into differential equations.

By multiplication, Lemma 61 generalizes to non-consecutive edges. Thus,

$$\frac{F_j(l)}{F_0(l)} \;\;=\;\; \frac{L'_j(l)}{l'} \;\;=\;\; L'_j(l) \tag{5.19}$$

holds. On the other hand, taking the derivative of the formula in Lemma 60 leads to

$$F'_j(l) \;+\; L'_{j-1}(l) \;\;=\;\; \cos\alpha\, L'_j(l). \tag{5.20}$$

We substitute in 5.20 both $L'_j(l)$ and $L'_{j-1}(l)$ by the expressions we get from 5.19 and obtain a linear differential equation for $F_j(l)$,

$$F'_j(l) \;-\; \frac{\cos(\alpha)}{F_0(l)} F_j(l) \;\;=\;\; -\frac{F_{j-1}(l)}{F_0(l)}.$$

The textbook solution for $y'(x) + f(x)y(x) = g(x)$ is

$$y(x) = \exp(-a(x)) \left(\int g(t) \exp(a(t))\, \mathrm{d}t + \kappa \right),$$

where $a = \int f$ and κ denotes a constant that can be chosen arbitrarily. In our case,

$$a(l) = \int -\frac{\cos(\alpha)}{A + \cos(\alpha)\, l} = -\ln(F_0(l))$$

because of $F_0(l) = A + \cos(\alpha)\, l$, and we obtain

$$F_j(l) = F_0(l)\Big(\kappa_j - \int \frac{F_{j-1}(t)}{F_0^2(t)}\,\mathrm{d}t\Big). \tag{5.21}$$

Next, we consider a linkage of ϕ-type, for parameters $l \in [l_1, l_2]$, and obtain analogously

$$\phi_j(l) = \phi_0(l)\Big(\lambda_j - \int \frac{\phi_{j-1}(t)}{\phi_0^2(t)}\,\mathrm{d}t\Big). \tag{5.22}$$

Now we determine the constants κ_j, λ_j such that the solutions 5.21 and 5.22 describe a contiguous curve. To this end, we must satisfy conditions 5.17 and 5.18.

We define $\kappa_0 := 1$ and

$$\kappa_{j+1} := \frac{\phi_j(l_2)}{F_0(0)} + \int \frac{F_j(t)}{F_0^2(t)}\mathrm{d}t|_{l=0}$$

so that 5.21 becomes

$$F_{j+1}(l) = F_0(l) \Big(\frac{\phi_j(l_2)}{F_0(0)} - \int_0^l \frac{F_j(t)}{F_0^2(t)}\,\mathrm{d}t\Big),$$

which, for $l = 0$, yields $F_{j+1}(0) = \phi_j(l_2)$ (condition 5.18).

Similarly, we set $\lambda_0 := 1$ and

$$\lambda_{j+1} := \frac{F_{j+1}(l_1)}{\phi_0(l_1)} + \int \frac{\phi_j(t)}{\phi_0^2(t)}\mathrm{d}t|_{l=l_1}$$

so that 5.22 becomes

$$\phi_{j+1}(l) = \phi_0(l) \Big(\frac{F_{j+1}(l_1)}{\phi_0(l_1)} - \int_{l_1}^l \frac{\phi_j(t)}{\phi_0^2(t)}\,\mathrm{d}t\Big),$$

and for $l = l_1$ we get $F_{j+1}(l_1) = \phi_{j+1}(l_1)$ (condition 5.17).

For simplicity, let us write

$$G_j(l) := \frac{F_j(l)}{F_0(l)} \quad \text{and} \quad \chi_j(l) := \frac{\phi_j(l)}{\phi_0(l)}, \tag{5.23}$$

which leads to

$$G_{j+1}(l) = \frac{\phi_0(l_2)}{F_0(0)}\chi_j(l_2) - \int_0^l \frac{G_j(t)}{F_0(t)}\,\mathrm{d}t \tag{5.24}$$

$$\chi_{j+1}(l) = \frac{F_0(l_1)}{\phi_0(l_1)}G_{j+1}(l_1) - \int_{l_1}^l \frac{\chi_j(t)}{\phi_0(t)}\,\mathrm{d}t. \tag{5.25}$$

Now we can make a useful observation. In order to find out if the fire fighter is ever successful, we need to check only the values of $F_j(l)$ at the end of each round.

Lemma 64 *The curve encloses the fire if and only if there exists an index j such that $F_j(l_1) \le 0$ holds.*

Proof. The free string shrinks to zero if and only if there exist an index j and argument l such that $F_j(l) \le 0$ or $\phi_j(l) \le 0$. Clearly, G_j and F_j have identical signs, as well as χ_j and ϕ_j do. Suppose that $G_j > 0$ and $G_{j+1}(l) = 0$, for some j and some $l \in [0, l_1]$. By 5.24, function G_{j+1} is decreasing, therefore $G_{j+1}(l_1) \le 0$. Now assume that $G_i > 0$ holds for all i, and that we have $\chi_{j-1} > 0$ and $\chi_j(l) = 0$ for some j and some $l \in [l_1, l_2]$. By 5.25 this implies $\chi_j(l_2) \le 0$, and from 5.24 we conclude $G_{j+1} \le 0$, in particular $G_{j+1}(l_1) \le 0$. $\qquad\square$

5.6.4   Recursions

Our next task is to make the integrals in 5.24 and 5.25 disappear by iterated substitution, and then, to substitute variable l with concrete values l_1 (resp. l_2), based on Lemma 64.

Let us define, for positive real numbers $x_0, \dots, x_n$, the auxiliary expression

$$I_n(x_n) \ := \ \int_0^{x_n} \frac{1}{F_0(x_{n-1})} \int_0^{x_{n-1}} \frac{1}{F_0(x_{n-2})} \dots \int_0^{x_1} \frac{1}{F_0(x_0)} \, \mathrm{d}x_0 \dots \mathrm{d}x_{n-1}.$$

By induction on n one quickly shows

$$I_n(x_n) \ = \ \frac{1}{n!} \frac{1}{\cos^n \alpha} \left(\ln\left(\frac{A + \cos \alpha x_n}{A} \right) \right)^n$$

since $F_0(x) = A + \cos \alpha \, x$; compare the discussion after formula 5.18. Now we iteratively substitute G_j in formula 5.24 and obtain

$$G_{j+1}(l) \ = \ \frac{\phi_0(l_2)}{F_0(0)} \sum_{\nu=0}^{j} (-1)^\nu \, I_\nu(l) \, \chi_{j-\nu}(l_2) \ + \ (-1)^{j+1} I_{j+1}(l). \tag{5.26}$$

By definition of l_1, according to formula 5.15, we have $\ln\left(\frac{A+\cos(\alpha)l_1}{A}\right) = 2\pi \cot \alpha$, so that setting $l = l_1$ in formula 5.26 leads to

$$G_j(l_1) \ = \ \frac{\phi_0(l_2)}{F_0(0)} \sum_{\nu=0}^{j} \frac{(-1)^\nu}{\nu!} \left(\frac{2\pi}{\sin \alpha}\right)^\nu \chi_{j-1-\nu}(l_2) \tag{5.27}$$

where, for convenience, $\chi_{-1}(l_2) := \frac{F_0(0)}{\phi_0(l_2)}$. We observe that this formula is also true for $j = 0$. Multiplying both sides by $F_0(l_1)$, and re-substituting 5.23, results in

$$F_j(l_1) \ = \ \frac{F_0(l_1)}{F_0(0)} \sum_{\nu=0}^{j} \frac{(-1)^\nu}{\nu!} \left(\frac{2\pi}{\sin \alpha}\right)^\nu \phi_{j-1-\nu}(l_2) \tag{5.28}$$

where $\phi_{-1}(l_2) := F_0(0)$.

In a similar way we solve the recursion in 5.25, using

$$\int_0^{x_n} \frac{1}{\phi_0(x_{n-1})} \int_0^{x_{n-1}} \dots \int_0^{x_1} \frac{1}{\phi_0(x_0)} \, \mathrm{d}x_0 \dots \mathrm{d}x_{n-1} \ = \ \frac{1}{n!} \frac{1}{\cos^n \alpha} \left(\ln\left(\frac{x_n}{l_1}\right) \right)^n$$

and $\ln\left(\frac{l_2}{l_1}\right) = \alpha \cot \alpha$, according to formulae 5.15 and 5.16. One obtains, after substituting $l = l_2$,

$$\phi_j(l_2) \ = \ \frac{\phi_0(l_2)}{\phi_0(l_1)} \sum_{\nu=0}^{j} \frac{(-1)^\nu}{\nu!} \left(\frac{\alpha}{\sin \alpha}\right)^\nu \hat{F}_{j-\nu}(l_1) \tag{5.29}$$

where $\hat{F}_0(l_1) := \phi_0(l_1)$ and $\hat{F}_{i+1}(l_1) := F_{i+1}(l_1)$.

Exercise 27 *Calculate $F_0(l_1)$, $\phi_0(l_1)$ and $\phi_0(l_2)$. Why are these values important for us?*

5.6.5 Generating functions

In order to solve the cross-wise recursions 5.28 and 5.29 for the numbers $F_j(l_1)$ we are interested in, we define the generating functions

$$F(Z) := \sum_{j=0}^{\infty} F_j \, Z^j \quad \text{and} \quad \phi(Z) := \sum_{j=0}^{\infty} \phi_j \, Z^j$$

where $F_j := F_j(l_1)$ and $\phi_j := \phi_j(l_2)$, for short. From 5.28 we obtain

$$F(Z) \;=\; \frac{F_0}{F_0(0)} \, e^{-\frac{2\pi}{\sin \alpha} Z} \left(Z \, \phi(Z) + F_0(0) \right), \tag{5.30}$$

and from 5.29,

$$\phi(Z) \;=\; \frac{\phi_0}{\phi_0(l_1)} \, e^{-\frac{\alpha}{\sin \alpha} Z} \left(Z \, F(Z) - F_0 + \phi_0(l_1) \right). \tag{5.31}$$

Both equalities can be verified by plugging in expansions of the exponential functions, using $e^W = \sum_{j=0}^{\infty} \frac{W^j}{j!}$, computing the products, and comparing coefficients. Now we substitute 5.31 into 5.30, solve for $F(Z)$, divide both sides by F_0 and expand by $e^{\frac{2\pi + \alpha}{\sin \alpha}}$ to obtain the surprisingly simple formula

$$\frac{F(Z)}{F_0} \;=\; \frac{e^{vZ} \;-\; r\,Z}{e^{wZ} \;-\; s\,Z}, \tag{5.32}$$

where v, r, w, s are the following functions of α:

$$v \;=\; \frac{\alpha}{\sin \alpha} \quad \text{and} \quad r \;=\; e^{\alpha \cot \alpha}$$

$$w \;=\; \frac{2\pi + \alpha}{\sin \alpha} \quad \text{and} \quad s \;=\; e^{(2\pi + \alpha) \cot \alpha}. \tag{5.33}$$

We recall that α denotes the angle of the fighter's velocity vector, given by $\alpha = \cos^{-1}(1/v)$.

It is possible to expand the inverse of the denominator in 5.32 into a power series. This leads to interesting expressions for the F_j; but how to derive their signs seems not obvious.

Exercise 28 *Show how to obtain 5.32 from 5.31 and 5.30.*

5.6.6 Singularities

Both, numerator and denominator of function $F(Z)$, are analytic on the complex plane. Thus, singularities of F can arise only from zeroes of the denominator,

$$e^{wZ} - sZ. \tag{5.34}$$

This equation has received some attention in the area of delay differential equations. We need a simple fact that has been published by Flajolet (1995).

Lemma 65 *For $s < ew$, equation 5.34 has an infinite, discrete set of conjugate complex zeroes none of which are real.*

As the fighter' speed v increases from 1 to infinity, angle α of her velocity vector increases from 0 to $\pi/2$. This causes the ratio s/w in 5.34 to decrease from infinity to zero. Exactly at $v_c = 2.6144\ldots$ does the equality $s/w = e$ hold. Hence, for $v > v_c$ we have $s < ew$, so that the denominator of $F(Z)$ has infinitely many strictly complex zeroes. The root classification given by Flajolet shows that only few of them can cancel out with the numerator of 5.32. A complete proof of this fact can also be found by Klein et al. 2014, Lemma 10-13.

We obtain the following from the above.

Lemma 66 *If $v > v_c$ then function $F(Z)$ has an infinite, discrete set of complex poles none of which are real.*

We are going to apply the following result from complex function theory; see, for example, Flajolet (2009).

Theorem 67 (Pringsheim) *Let $H(Z) = \sum_{n=0}^{\infty} a_n Z^n$ be a power series with finite radius of convergence, R. If $H(Z)$ has only non-negative coefficients a_n, then point $Z = R$ is a singularity of $H(z)$.*

Now we are ready to prove Theorem 59.

Proof.[Proof of Theorem 59] Suppose that the fighter's speed v is larger than $v_c \approx 2.6144$. By Lemma 66, $F(Z)$ does have a discrete set of poles, and therefore, a finite radius of convergence, R. If all coefficients F_j of $F(Z)$ were positive, R would be a singularity of $F(Z)$, by Pringsheim's theorem; but we know from Lemma 66 that there are no real singularities. Thus, there must be coefficients $F_j \leq 0$, and we conclude from Lemma 64 that the fighter succeeds in containing the fire.   □

5.7   General Constructions: Lower and upper bounds

Now we consider the case that the firefighter has some speed v but is able to distribute the speed for the construction of more than one firebreak at the same time. In some sense this would mean that the firefighter are also able to jump from one position to the other. The firebreak can be constructed everywhere outside the spreading fire as long as the overall speed is not exceeded.

Due to our considerations above there is a simple upper bound $v > 2$ for a spreading fire circle. We simply construct to symmetric logarithmic spirals along the boundary of the fire with excentricity $v/2 = \frac{1}{\cos \alpha}$ for $v/2 > 1$ and guarantee to enclose the fire in any case as depicted in Figure 5.22.

There is a lower bound of $v \geq 1$ which already makes some effort as we will see below, For the interval $v \in (1, 2)$ it is still an open question whether there is a strategy for v or not.

Theorem 68 *For any speed $v > 2$ there is a successful general strategy that encloses any spreading fire circle. For speed $v \leq 1$ there is no such general strategy.*

Proof. The upper bound was shown above as depicted in Figure 5.22. For the lower bound we choose $v = 1$ and assume that there is a successful strategy S. The strategy S will finally enclose the spreading fire. Let x denote the final *stone* for the enclosement at some time t_x. There is a situation as depicted in Figure 5.23. There will be some outer connected boundary firebreak S_O which consists of a *ring* R_O and some simple tree paths S_O^j which are connected with R_O. Other tree like connected paths, S_I^i, constructed by S are not connected to the outer boundary. They serve as additional obstacles for slowing down the spread of the fire. We can assume that finally at time t_x the strategy builds the first overall loop. So all S_I^i and S_O^j are tree like paths.

Let Π_s^x denote the geodesic shortest path from the source of the fire to the point x under the presence of the obstacles R_O and S_O. Assume that this path crosses n inner paths $S_I^{j_i}$ for some subset $\{j_1, j_2, \ldots, j_n\}$. For the contradiction we can omit these obstacles by the following argument. Any connected tree like obstacle path $S_I^{j_i}$ that intersects with Π_s^x has a starting point s_{i_j} and a leaving point t_{i_j} w.r.t. the orientation of Π_s^x from s to x. Following the boundary from s_{i_j} to t_{i_j} in clockwise and conuterclockwise direction along $S_I^{j_i}$ gives two path with length twice
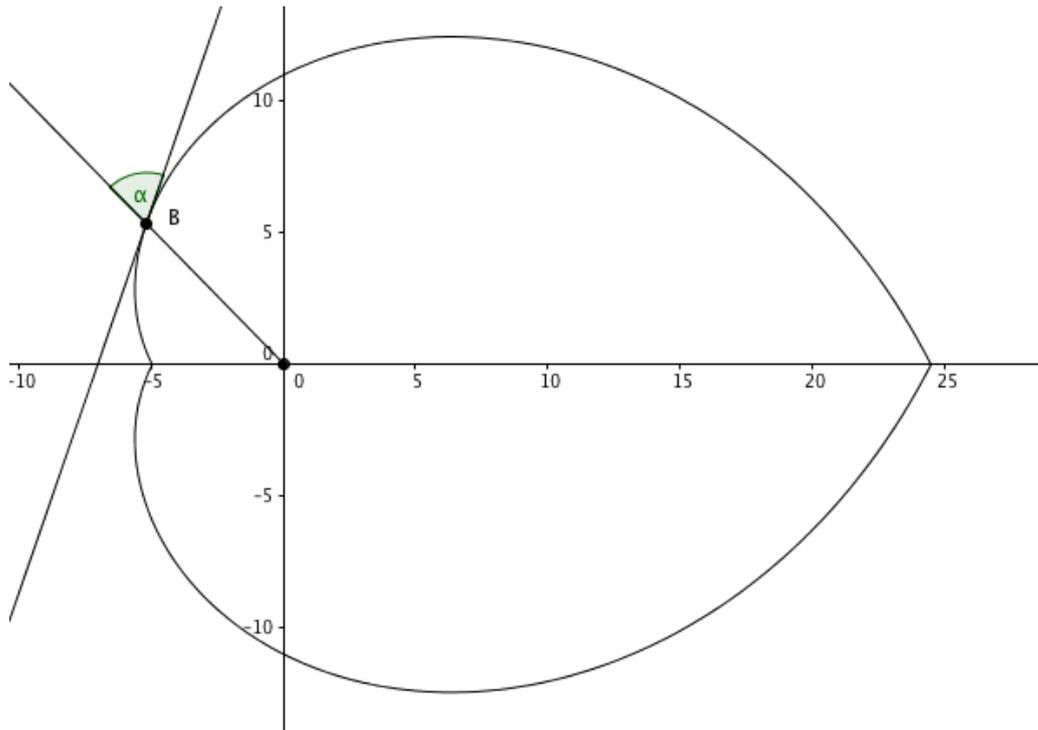
Figure 5.22: For $v > 2$ the firefighter simultaneously build two spiral of excentricity α for $v/2 = \frac{1}{\cos \alpha}$ at the boundary of the fire. The strategy is successful for any such speed $v > 2$.
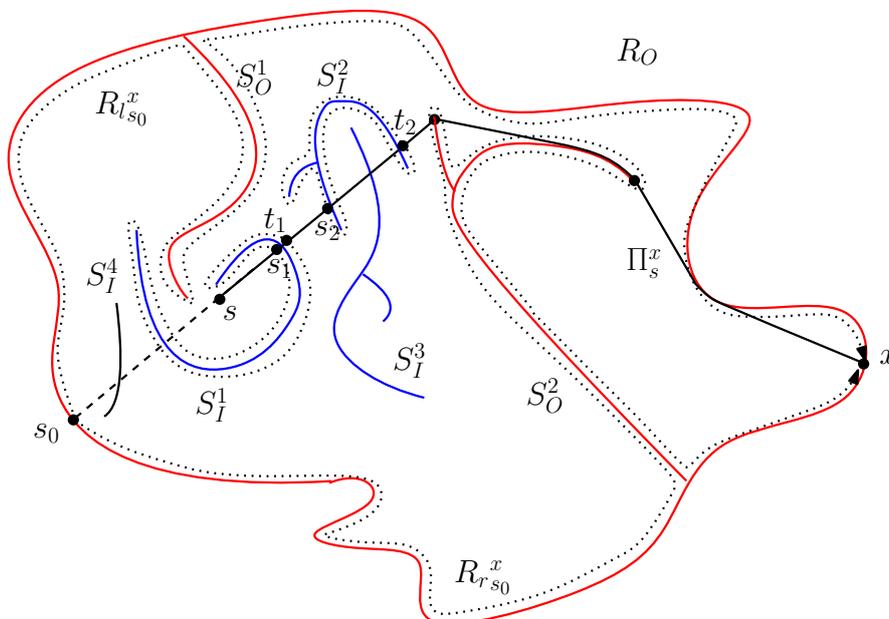

Figure 5.23: Asume that for $v = 1$ the spreading fire starting from s is finally enclosed at x at time t_x. The inner obstacles $S_I^{j_i}$ can be omitted. The fire will reach the final *stone* x along Π_s^x before all obstacles can be constructed. This means that there is no such construction!

the length of $S_I^{j_i}$, the shorter of these two path has lenght not larger than $S_I^{j_i}$. Therefore, during the construction of $S_I^{j_i}$ the fire can move form s_{i_j} to t_{i_j}. This holds because the construction was done with the same speed $v = 1$. By this argument we simply omit all inner obstacles $S_I^{j_i}$.

We prolong the path Π_s^x back to the outer boundary to some point s_0 on R_O and S_O as depicted in Figure 5.23 such that $\Pi_{s_0}^x$ is a shortest path from s_0 to x under the presence of R_O and S_O. The sum of the path length of the outer boundary exceeds the length of $\Pi_{s_0}^x$, because in the worst case the path $\Pi_{s_0}^x$ can only follow the full paths R_O and S_O. Thus, before x can be placed the fire has reached the point x along the path $\Pi_{s_0}^x$ which is a bit shorter than $\Pi_{s_0}^x$. The outer boundary cannot be closed at point x. $\square$

Exercise 29 *Show that in the above lower bound construction it is allowed that we can assume that no loop obstacles (apart from R_O) have been constructed.*

The above proof follows an idea of Bresson et al. They brought up the question whether it is possible to show that the lower bound lies inside $v \in (1, 2)$ or is indeed $v = 2$. This is still an open question.

Exercise 30 *Show that in the above proof there is not much room for improving the lower bound to $v > 1$. Try to construct a corresponding example.*

Chapter 6

Escape Paths for the Intruder

In this chapter we would like to discuss a reverse situation. An intruder tries to escape from an environment as soon as possible. This work is inspired by the famous question of Bellman (brought up in 1956) who asked for the shortest escape path from an unknown forest.

As before we would like to consider geometric variants as well as more discrete situations. Again, the problem statement can be considered to be a game. The intruder has some abilities and tries to escape from the environment quickly whereas the adversary can manipulate the environment so that the intruder leaves the environment very late. We are looking for apropriate escape strategies for the intruder and consider different performance measures.

6.1 Lost in a forest

Assume that a simple region R in the plane is given which boundary is formally defined by a closed Jordan curve B that divides the plane in two simply connected regions. The intruder is located inside R and tries to find the boundary B as soon as possible by a deterministic escape path in the plane. We assume that the intruder has no sight system and only detects the boundary by touching it. The starting position p inside R and the rotation of R is unknown for the intruder but the exact shape of R is known. For example, somebody is located inside a dark forest of known shape and tries to get out of the forest as soon as possible. The deterministic escape path Π has to lead out of the region R for any starting position $p \in R$ and any rotation of R around p.

The performance measure for the path Π is simply its length. There will be a worst case starting position p of Π and a worst case rotation of R so that for p the full path length of Π is required to hit the boundary B. If no such point exists, there will be a better path Π of shorter length.

Considering such escape paths has a long tradition as mentioned above. Unfortunately, the optimal escape paths is only known for some very special shapes and totally unknown for general (polygonal) environments. We first discuss some simple convex situations where the diameter is optimal.

Conversely, the problem can also be considered as a covering problem. Consider the class C_L of rectifiable curves in the plane of some lenght L. Find an environment R of *small size* that can be rotated and translated so that it covers any curve of C_L. In the literature for $L = 1$ such covers are also denoted as *worm covers*. So we are searching for environments that are worm covers and but at least one worm finally touches the boundary from any starting situation.
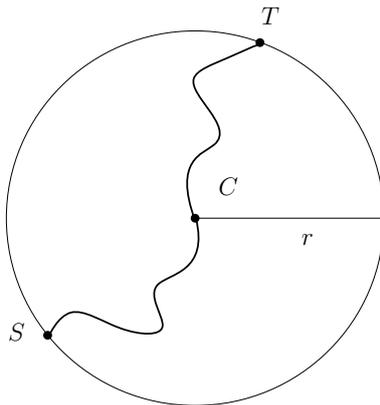
Figure 6.1: A cirlce of radius r covers any path of lenght $< 2r$. The diameter is a shortest escape path.

6.1.1   Simple examples and the diameter path

Let us assume that R is a circle of radius r.

Theorem 69 *The shortest escape path from a circle R of radius r is given by the diameter segment of R. Conversely, a circle of radius $\frac{1}{2}$ is a worm cover.*

Proof. Let us assume that the optimal escaps path Π of lenght L is given. We consider the point C on L that splits Π into two parts of lenght $L/2$. Let S denote the starting point of Π and T denote its endpoint. Now place C to the center of the circle; see Figure 6.1. In order to leave the circle at least S or T has to touch the boundary of R, thus $|CT| \geq r$ or $|CS| \geq r$. on the other hand the diameter d of R of lenght $2r$ is an escape path.   □

Interestingly, also the semicircle has the same escape path and a semicirlce of diameter 1 is also a worm cover. The proof is a bit more complicated and was given by A. Meir and manifested by Wetzel (1973).

Theorem 70 *The shortest escape path of a semicircle R of radius r and diameter $2r$ is given by the diameter. Conversely, a the semicircle of radius $\frac{1}{2}$ is a worm cover.*

Proof. Let us assume that the escape path is a path with start end endpoint S and T. We rotate the path so that ST is in parallel with the base line B_l of the semicircle and we also translate the segment (and the path) so that there is a single tangent point I on the base B and all other points of the path Π lie above B_l as depicted in Figure 6.2. Consider an arbitrary point $X \in Pi$, w.l.o.g. we assume that X lies inside the path from S to I. We also consider the reflections of S' and T' of S and T along the base line B_l. The segment ST' intersects the base line at some point O. The length of the segment SX is shorter than Π_S^X. By refeflection the length of the segment XT' is shorter than the path Π_X^T.

Now we translate the construction so that O is the center of the semicircle. We would like to argue that $|XO| \leq r/2$ holds. This means that any X is inside the semicircle which gives the conclusion.

We consider the triangle SXT' where O divides ST' into two parts of the same length. By geometry we know that the *median XO* is of the triangle is shorter that $\frac{1}{2}$ the length of the adjacent sites.

This means that

$$|XO| \leq \frac{1}{2}(|XS| + |XT'|) \leq \frac{1}{2}(\Pi_S^X + \Pi_X^T) < r\,.$$
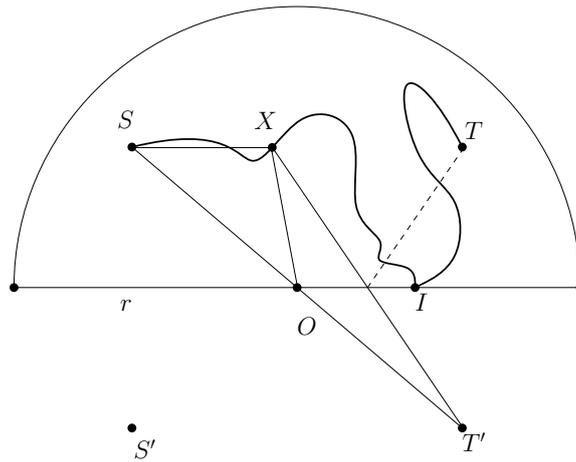
Figure 6.2: A semicirlce of radius r covers any path of lenght $< 2r$. The diameter is a shortest escape path.

$\square$

Exercise 31 *Show that the* median *of the triangle is always shorter than the average (factor $\frac{1}{2}$) of the length of the adjacent sites.*

Exercise 32 *Show that the above Theorems also hold for closed paths. Note that we considered open paths in the proofs above.*

In general for a given region R the diameter d is defined to be the longest shortest path between two points in R. The corresponding points are always located on the boundary of R (otherwise there exist two points connected by a longer shortest path). Thus, any diameter (path) is always an escape path, it need not be an optimal escape path as we will see in the next section. But for some convex objects R the diameter is indeed optimal as we will prove now. Some fatness condition is required.

First we consider the rhombus R of diameter L and angle $\theta = 60\circ$ as depicted in Figure 6.3. We would like to show that any escape path has lenght at least L. As already mentioned the diameter of lenght L is an escape path. The following proof stems from Poole and Gerriets (1973).

Theorem 71 *The optimal escape path for the rhombus R_α of diameter L and angle $\alpha = 60°$ is given by its diameter.*

Proof. As in the previous proofs we split an optimal escape path of some lenght $L' < L$ into two halfs of lenght $L'/2$ and consider the mean point C. We let C slide along the shorter diagonal BE as shown and rotate the path so that the path Π_S^C is tangent to AB and the path Π_C^T is tangent to BD. Such a rotational center for $C \in BE$ and orientation of R always exists. (Note that in an extreme case C could be located at B and only touches both AB and BD.) Let $X \in AB$ and $Y \in BD$ denote the corresponding tangent points of Π_S^C and Π_C^T, respectively.

Because L' is covered by R at least one path Π_S^C or Π_C^T has to hit the upper angle AED of R. W.lo.g. assume that Π_C^T hits AED. Consider the shortest path from Y to AED and to BE which meet ED at p_1 and BE at p_2 by angle $\pi/2$, respectively. This means that Π_C^T cannot be shorter than the sum of lenghts of p_1Y and p_2Y.
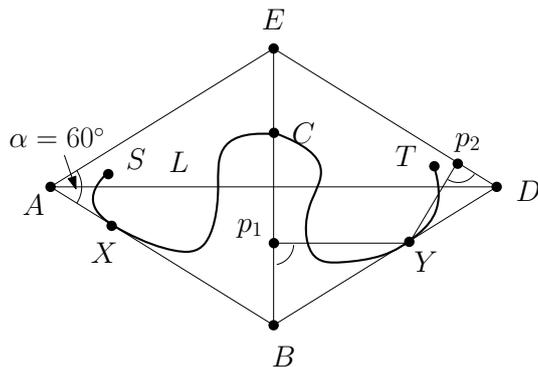
Figure 6.3: The rhombus R_α of diameter L and angle $\alpha = 60°$. The diameter is the shortest escape path since $|p_1 Y| + |p_2 Y|$ equals $L/2$ for any point Y on BD.
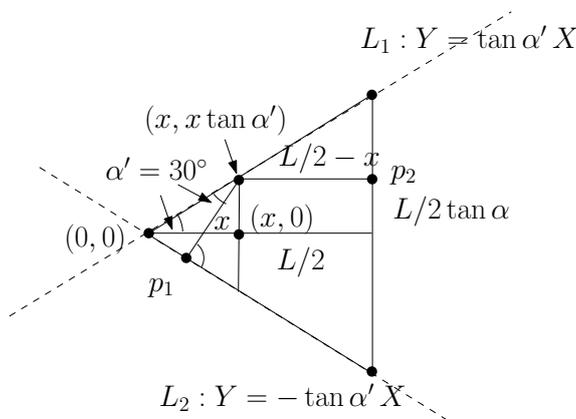

Figure 6.4: Parameterization for the conclusion in Theorem 71. The distance from $(x, x \tan \alpha')$ on L_1 to p_1 on L_2 also equals x for $\alpha' = 30°$.

Finally, we show that $|p_1 Y| + |p_2 Y|$ equals $L/2$ for all $Y \in BD$, which concludes the proof. This holds by the geometric arguments shown in Figure 6.4.   □

Exercise 33 *Is the shortest escape path always unique? Answer the question for convex and non-convex regions R.*

6.1.2   Besicovitch Zig-Zag path

Up to now all shortest escape paths where given by the diameter of the given convex object. This is not always true as we will show by a family of isosceles triangles T_α with base of length

$$b_\alpha = \sqrt{1 + \frac{1}{9 \tan^2 \alpha}} \, .$$

as shown in Figure 6.5 for $\alpha = 60°$ where T_α is a unilateral triangle. We will finally show that we can escape from the unilateral triangle of side-length 1 by a symmetric Zig-Zag path of lenght $\sqrt{\frac{27}{28}} < 1$, although 1 is obviously its diameter.

In general we consider α to be in the interval from roughly $52.24°$ up to $60°$. For $\alpha > 60°$ the base b_α is no longer the largest segment (and not the diameter any more), the reason for $\alpha \geq 52.24°$ is shown below. We show that for any T_α the shortest symmetric Zig-Zag-Path is

an escape path for T_α with path lenght smaller than b_α. More precisely, the shortest symmetric Zig-Zag path will have lenght 1 and leaves T_α from any starting point.

The following result goes back to Coulton and Movshovich (2006). First, we define a symmetric Zig-Zag escape path. We orient T_α so that the base b_α runs in parallel to the X-axis and runs from $(0,0)$ to $(b_\alpha, 0)$. The remaining segments l_α and r_α of T_α run above the X-asis in parallel along the lines $L_1 : Y = \tan \alpha X$ and $L_2 : Y = \tan \alpha (b - X)$, respectively as given in Figure 6.5.

The symmetric Zig-Zag consists of three consecutive segements of the same length and starts at the origing of T_α. Any segment has the same altitude h w.r.t. the base b_α. The last segment exactly touches the segment r of T_α. By construction any such path is an escape path for the corresponding T_α.

Now, we would like to construct a Zig-Zag path of length 1 for any T_α such that the path is the shortest symmetric Zig-Zag for a corresponding b_α. Such a path is the shortest, if its *straightened path* of the same length hits the line $L : Y = 3 \tan \alpha (b_\alpha - X)$ by a right angle as depicted in Figure 6.5 i). By congruence as shown in Figure 6.5 ii) we conclude that $\frac{1}{x} = \frac{b_\alpha}{1}$ which gives $x = \frac{1}{b_\alpha}$. Finally we determine b_α by $y = \tan \alpha \left(b_\alpha - \frac{1}{b_\alpha} \right)$ and $x = \frac{b}{1}$ and $x^3 + (3y)^2 = 1$ which gives

$$b_\alpha = \sqrt{1 + \frac{1}{9 \tan^2 \alpha}} \, .$$

For $\alpha = 60°$ we have $b_\alpha = \sqrt{\frac{28}{27}}$ and we can escape from the equilateral triangle of side-length 1 by a symmetric Zig-Zag path of lenght $\sqrt{\frac{27}{28}} < 1$, although 1 is obviously its diameter.

For small α there might be other three-segment paths that also have distance 1 or even a shorter distance. This can happen for example, if a line $L_3 : Y = \tan(2\alpha)$ runs in parallel with L_2 as shown in Figure 6.6. This means $-3 \tan \alpha = \tan 2\alpha$ or $\tan \alpha = \sqrt{\frac{5}{3}}$. To avoid such situations we require $\alpha \geq \alpha_0$ where α_0 solves the equation.

Theorem 72 *For any $\alpha \in [\arctan(\sqrt{\frac{5}{3}}), 60°]$ there is a symmetric Zig-Zag path of lenght 1 that is an escape path of T_α smaller than the diameter b_α.*

Before we give a proof for the fact that the corresponding symmetric Zig-Zag escape paths are indeed optimal escape paths for any T_α we first introduce some other models and interpretations of the problem.

6.2 Different models and cost measures

Up to now we have considered the case that the intruder tries to escape from a geometric environment without any knowledge of its position. Let us assume that a bit more information is given and let us also consider a somewhat more discrete version. The intruder starts at the source of m long corridors, each of which finally lead out of the environment. The agent also knows the depth s_i but not the correspondance to the corridors.

Consider the situation where a set L_m of m line segments s_i of unknown length $|s_i|$ (which might represent dark corridors) are given and an agent has to find the end of only *one* arbitrary corridor as depicted in Figure 6.7(i). Just choosing a single corridor and move to its end might be very bad, if it is unfortunately the largest corridor. So in this escape problem the agent will move into one corridor s_{j_1} up to a certain distance x_1 and then check another corridor s_{j_2} for another distance x_2 and so on. Finally he will hit the end of one of the corridors with hopefully overall short path length.
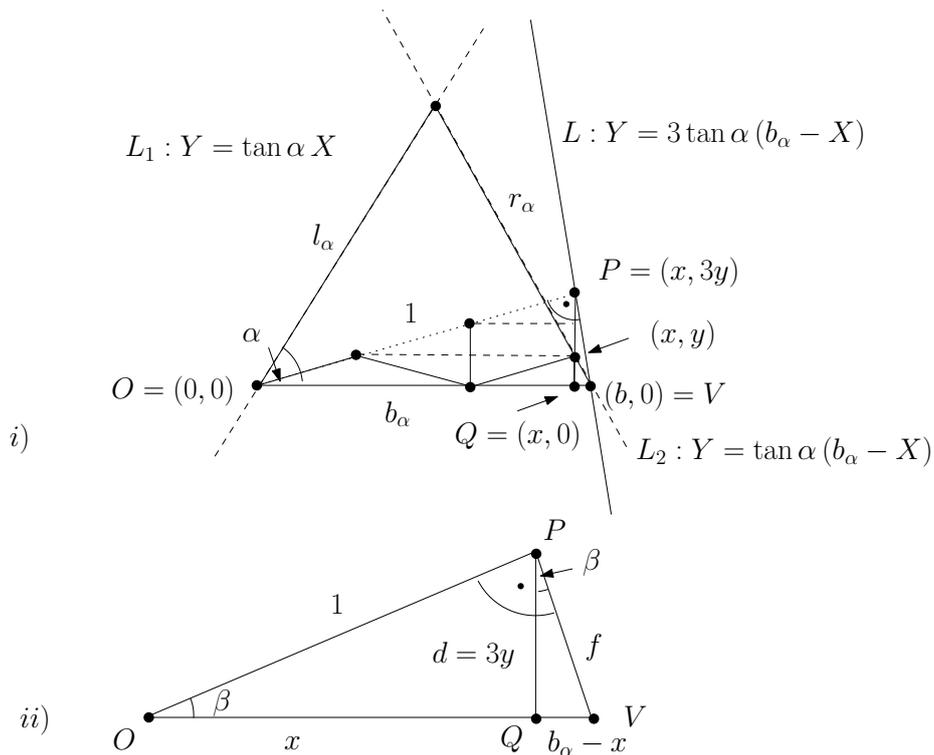
Figure 6.5: i) Any symmetric Zig-Zag path for T_α is an escape path. For the shortest such path the straightened segment mets the line $L : Y = 3 \tan \alpha (b_\alpha - X)$ by a right angle. ii) Using the congruent triangles OPQ and OPV we have $\frac{1}{x} = \frac{b}{1}$.
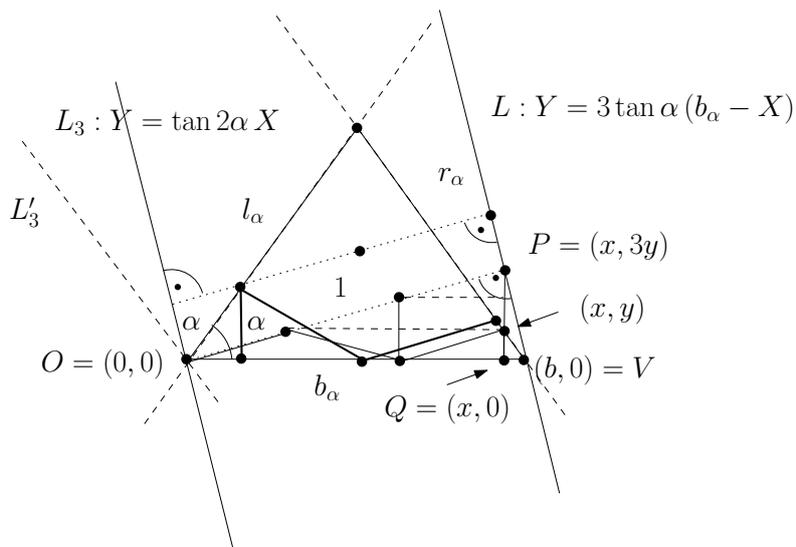

Figure 6.6: If α is too small, other Zig-Zag path might have a better performance (lenght ≤ 1).
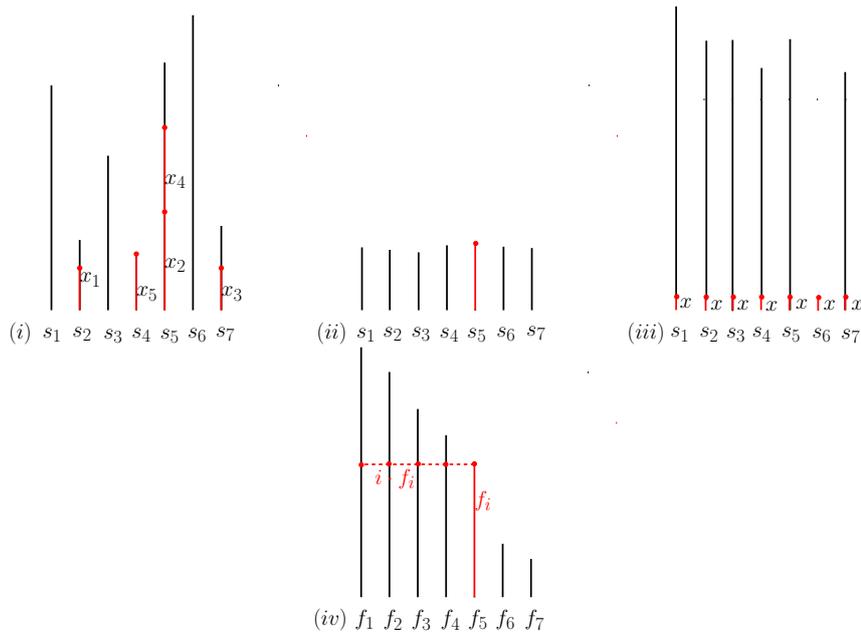
Figure 6.7: (i) Online searching for the end of one segment (or digging for oil) for $m = 7$ segments of unknown length as considered by Kirkpatrick. A reasonable strategy changes successively from one segment to the other with distances x_i on s_{j_i}. It is allowed to resume searching at the previous end-point with no extra cost (see x_2 and x_4 on $s_{j_2} = s_{j_4} = s_5$). The strategy reaches the end of segment s_5 after 5 movements $(x_1, \ldots, x_5)$. If at least the distance distribution is known, the problem is easier to solve. There are two extreme cases. (ii) If all segments have almost have the same length, it is reasonable to move along an arbitrary segment with largest distance, this is almost optimal. (iii) If there is one segment of very short length and all other segments are very long, one will find the end of the short segment by checking all segments with the shortest distance. This path is also short. (iv) In general the *discrete certificate* is defined for the given distance distribution. If $f_1 \geq f_2 \geq \cdots \geq f_m$ is the order of the length of the segments, it is always sufficient to check i arbitrary segments with length f_i and $\min_i i \cdot f_i$ is the best such strategy.

Kirkpatrick (2009) introduced this problem and also motivates the situation by the scenario of digging for oil at m locations s_i where the distance $|s_i|$ to the source of the oil of place s_i is not known. It is sufficient to get to the source of one place and the overall effort should be small. In this scenario it is allowed to resume the movement (or digging) for a location s_i at the endpoint where s_i was left at the previous visit; see Figure 6.7(i). There are no extra costs for moving to the previously reached depth at s_i.

Now we define a performance measure. Let us assume that all m distances $|s_i|$ are known but not the correspondence to the places s_i. In this case we can sort the m distances and obtain a discrete distance distribution of the length of the segments.

First, consider the extreme situations in Figure 6.7(ii) and (iii) If all segments s_i almost have the same length, a successful strategy will simply use the maximal length x among all segments, checks this for an arbitrary segment and will succeed with path length x in the worst case; Figure 6.7(ii). If on the other extreme (see Figure 6.7(iii)) the distance to a single source is small but the distances are very large to all others (and we only know this distribution), the best option is to check all segments successively by the small distance x. This gives an effort of at most $x \cdot m$ in the worst case when the small segment is found at the latest visit.

In general Kirkpartrick defines a *certificate* that takes the distance distribution into account.

The segments are sorted by distance s_i. Let f_j with $j = 1, \ldots, m$ denote the sorted list of the lengths of the segments with decreasing distances f_j as shown in Figure 6.7(iv). Now for distance f_i there are exact i segments that are larger than or equal to f_i. Thus, after checking arbitrary i segments s_j with distance f_i it is clear that we will find the end of at least one segment. In the worst case the i largest segments have been checked including f_i. We summarize the above ideas in the following Theorem.

For a set L_m of m line segments s_i and a set of F_m of m length f_j, there is a permutation π so that $|s_i| = f_{\pi(i)}$ holds. Only L_m and F_m are given, the permutation is unknown. The agent can make use of a startegy as shown above. For any strategy A there will be a worst-case permutation $\pi(A)$ so that the traveling cost for finding at least one entrance is maximal.

For convenience it suffices to consider that only F_m is given and let us further assume that F_m is sorted, that is $f_1 \geq f_2 \geq \cdots \geq f_m$. Let $C(F_m, A)$ denote the maximum travel cost for algorithm A and list F_m which is attained for some permutation $\pi(A, F_m)$. For a given set F_m we are searching for the best strategy A and define the maximum-traversal-cost, $\mathrm{maxTrav}(F_m)$ by the minimum cost over all possible traversal strategies A, that is

$$\mathrm{maxTrav}(F_m) := \min_A C(F_m, A)\,.$$

Theorem 73 *For a set of sorted distances F_m (i.e. $f_1 \geq f_2 \geq \cdots \geq f_m$) we have*

$$maxTrav(F_m) := \min_i i \cdot f_i\,.$$

Proof. Consider an arbitrary strategy A. If a strategy A moves less than $\min_i i \cdot f_i$ in total, the strategy has moved less that $j \cdot f_j$ for $j = 1, \ldots, m$. Thus, for any j, the strategy has visited less than j segments, say $k_j < j$ segments, up to distance f_j. Let us assume that the strategy has visiting depth $d_1 \geq d_2 \geq \cdots \geq d_m$ for the corridors. The adversary choose a permutation $\pi(A, F_m)$ so that the j-largest visiting depth d_j of A is applied to the segment of depth f_j. Thus the strategy has not checked the segment for f_1 up to the end, since $1 \cdot f_1$ has not been traveled in total. The segments of depths f_1 and f_2 has also not been visited up to the end since $2 \cdot f_2$ has not been travelled in total and the visiting depth at f_1 is as least as large as the visiting depth in f_2. Successively, the segments of depths $f_1, f_2, \ldots f_j$ has not been visited up to the end, since $j \cdot f_j$ has not been travelled and the visiting depths of $f_1, f_2, \ldots f_{j-1}$ are as least as large as the visiting depth of f_j. Altogether, the strategy is not successful for π with cost less than $\min_i i \cdot f_i$.

On the other hand if a strategy visits exactly i segments with depth f_i in the worst case the adversary presents the largest corridors but the ith-largest has length f_i. This means by $\min_i i \cdot f_i$ the strategy will be successful for any permutation. □

Now let us assume that in the online sense the distances are totally unknown to the agent. Kirkpatrick defines a general strategy that always approximates the certificate whithin a factor of $O(\mathrm{maxTrav}(F_m) \log(\min(m, \mathrm{maxTrav}(F_m))))$ for any list F_m of m totally unknown segments. It is shown that this factor is tight. The corresponding *dovetailling* strategy subdivides the overall digging length successively in a logarithmic way among an arbitrary order of the segments.

The online algorithm subdivides the cost w.r.t. a (discrete) hyperbolic function on the corresponding m segments. More, precisely the algorithm work in rounds $c = 1, 2, 3, 4, \ldots$ For any round c from left to right the path length on segment i is extended up to distance $\left\lfloor \frac{c}{i} \right\rfloor$. Figure 6.8 shows the extension scheme after 11 rounds for $m = 9$ segments. In any round any segment is extended at most by a distance of 1. If some segment i is extended in step c we have $\frac{c}{i} = t$ and the segments $1, 2, \ldots, i$ are all visited with depth at least t. This means that for any f_i in the interval from $t - 1$ to t, the strategy covers i segments up to distance f_i after step c for any i with $\frac{c}{i} = t$.
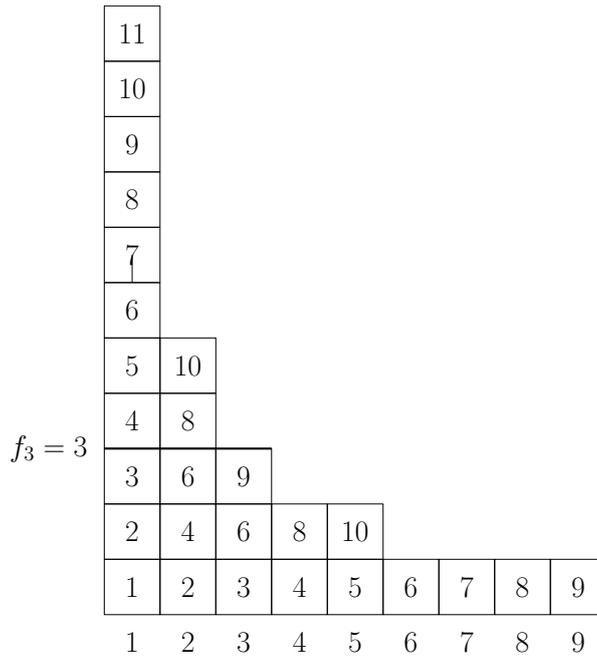
Figure 6.8: For $m = 9$ segments, in step $c = 1, 2, \ldots$ segment i is extended up to distance $\left\lfloor \frac{c}{i} \right\rfloor$ from left to right. For $f_3 = 3$, 3 segments will be covered with distance 3 during execution of step 9 on segment 3.

For convenience we assume that F_m contains only integer values. Now assume that for F_m we have $\min_i i \cdot f_i = j \cdot f_j$. There is some c with $c = j \cdot f_j$ and the strategy will be successful in this step. For this c the overall cost for the strategy is

$$\sum_{t=1}^{m} \left\lfloor \frac{c}{t} \right\rfloor \leq \sum_{t=1}^{\min(m,c)} \frac{c}{t} \leq c + \int_{1}^{\min(m,c)} \frac{c}{t} \, dt = c(1 + \ln \min(m, c)) \, .$$

Altogether, the following Theorem holds.

Theorem 74 *The hyperbolic traversal algorithm solves the multi-segment escape problem for any list F_m with maximum traversal cost bounded by*

$$D \cdot (maxTrav(F_m) \ln(\min(m, maxTrav(F_m))))$$

for some constant D.

Proof. By the considerations above for F_m with integer values for $D = 2$. □

Exercise 34 *Show that Theorem 71 also holds for non-integer F_m by doing the full analysis with a corresponding D.*

Finally, we would like to argue that any deterministic strategy can be forced to have cost in the size of $d \cdot (maxTrav(F_m) \ln(\min(m, maxTrav(F_m)))$ and the corresponding dovetailing strategy is optimal in this sense. Therefore we assume that we consider overall cost C and define family of sets $F'_m(C)$ by $f'_i(C) = \frac{C}{i}$, so that $maxTrav(F_m(C)) = C$ and also $\sum_{i=1}^{m} f_i(C) := d \cdot C \ln \min(C, m)$ hold, by choosing d apropriately.

For large C we have situation as depicted in Figure 6.9. As long as the overall cost of a deterministic strategy A does not exceed cost $d \cdot C \ln \min(C, m)$ we do not let any corridors end.

$$d_1 + \epsilon/m =: f_1(A, C)$$

$$f_1'(C) = \frac{C}{1}$$

$$f_2(A, C) := d_2 + \epsilon/m$$

$$f_3(A, C) := d_3 + \epsilon/m$$

$$d_4 < f_4'(C) = \frac{C}{4}$$

$$f_7(A, C) := f_7'(C)$$

$$f_8(A, C) := f_8'(C)$$

1   2   3   4   5   6   7   8
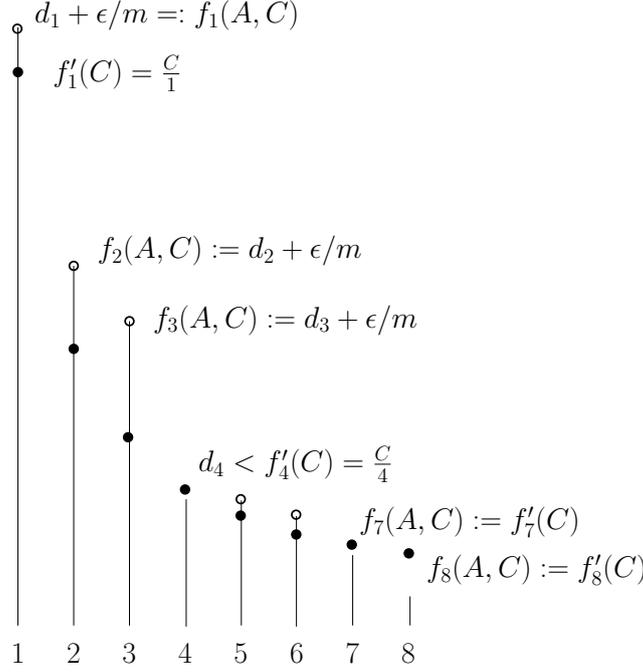
Figure 6.9: For $m = 8$ segments and some fixed C we define the lengths $f_i'(C) = \frac{C}{i}$ and let d be a constant so that $\sum_{i=1}^{8} f_i'(C) := d \cdot C \ln \min(C, m)$. A strategy A has visited the segments up to distance $d_1 \geq d_2 \geq \cdots \geq d_8$ and now attains the cost $\sum_{i=1}^{m} d_i \geq d \cdot C \ln \min(C, m)$ for the first time. For any segment j with $d_j < f_j'(C)$ we set $f_j(A, C) := f_j'(C)$. For any k with $d_k \geq f_k'(C)$ we set $f_k(A, C) := d_k + \epsilon/m$ for some arbitrarily small ϵ. There has to be some smallest index i so that $d_i < f_i'(C)$ and $i \cdot f_i'(C) = C$ is an upper bound of the optimal cost for $F_m(A, C)$.

For increasing visiting depth $d_1 \geq d_2 \geq \cdots \geq d_m$ of the strategy we consider the first moment in time where $\sum_{i=1}^{m} d_i \geq d \cdot C \ln \min(C, m)$ is attained. Immediately before not all corridors could have have been visited up to distance $f_i'(C)$, otherwise the overall path length $d \cdot C \ln \min(C, m)$ has been attained. Consider the smallest index i so that $d_i < f_i'(C)$. We finish the situation by fixing segment k with $d_k \leq f_k(C)$ to its final lenght $f_k(A, C) := f_k'(C)$ and fix all remaining segments with $d_j > f_j'(C)$ by $f_j(A, C) := d_j + \epsilon/m$ for some arbitrarily small ϵ. Note that $f_k(A, C), f_j(A, C) < f_i(A, C)$ holds for all corresponding $k, j > i$. The corresponding configuration can be denoted by $F_m(C, A)$. This means that $\mathrm{maxTrav}(F_m(C, A)) \leq i \cdot f_i = \mathrm{maxTrav}(F_m'(C)) = C$ holds and the lower bound is completed.

Theorem 75 *For any deterministic online strategy A that solves the multi-segment escape problem we can construct input sequences $F_m(A, C)$ so that A has cost at least $d \cdot C \ln \min(C, m)$ and $\mathrm{maxTrav}(F_m(C, A)) \leq C$ holds for some constant d and arbitrarily large values C.*