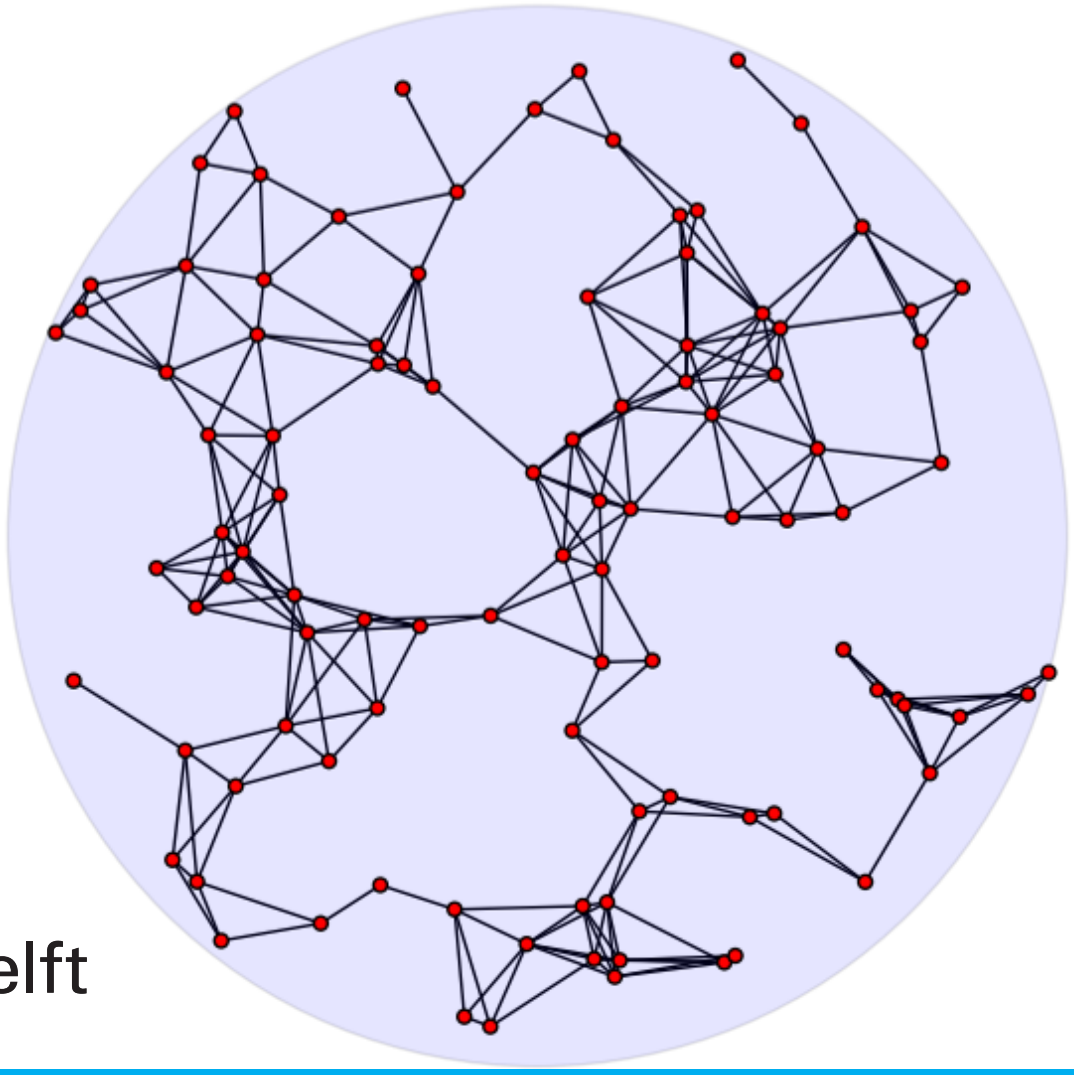
Firefighting! Strategies for Random Geometric Graphs using an Analytic Approach

Bachelor thesis

Guus Gelderblom



TU Delft

Firefighting! Strategies for Random Geometric Graphs using an Analytic Approach

Bachelor thesis

by

Guus Gelderblom

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended on Thursday June 26, 2025 at 1:30 PM.

Student number:	5877989	
Project duration:	April 22, 2025 – June 26, 2025	
Supervisors:	Dr. J. Komjáthy,	TU Delft
	Y. Moreno Alonso,	TU Delft
Thesis committee:	Dr. J. Komjáthy,	TU Delft, supervisor
	Dr. A. B. T. Barbaro,	TU Delft

TU Delft

Laymen's summary

In this bachelor thesis we find strategies for fighting fire in forests. As in a rainforest we assume that trees are placed randomly, since they are not planted by men. It is very well imaginable that a tree on fire cannot ignite a tree that is very far away. However, it can ignite the trees around it. We make a network of trees connected to each other if fire can spread from one to the other. We put firefighters on trees to prevent the fire from spreading to those trees. By placing the firefighters in a specific way we can contain the fire such that it no longer spreads. We explain how we can build a boundary of firefighters in a teardrop-shape to contain the fire. We also look at what happens when fire is influenced by wind. Fire propagates faster in the direction of the wind and slower in opposite direction. We prove that we can still contain a specifically shaped fire under the influence of wind when we use the same strategy as before.

Summary

In this bachelor thesis, we investigate some strategies for the firefighting problem on the random geometric graph. The firefighting problem is a problem on graphs where fire breaks out on a set of vertices. Each subsequent turn we can protect vertices using firefighters, after which the fire spreads to all unprotected neighbours of vertices on fire. The process ends when the fire cannot spread anymore. The protected vertices and the vertices not on fire are the vertices saved. A random geometric graph is a graph of which the vertices are obtained by a homogeneous Poisson Point Process of intensity λ. Vertices are connected by edges if they are distance at most 1 apart. We explain the proof from [3] that we can bound the number of firefighters needed to contain the fire from above by 2λ. We do so by translating the firefighting problem to the problem of surrounding a continuously growing blob with a fence built at rate ρ. We explain the proof from [3] that the blob can be surrounded by a teardrop-shaped fence built at rate $\rho > 2$ and that the blob cannot be surrounded by a fence built at rate $\rho \leq 1$. Finally, we consider a *blob with wind*. In the real world fire propagation is influenced by wind. We therefore assume that the blob grows faster in the direction of the wind, we call this region the fire front and slower in opposite direction, obtaining a blob with wind. We find that for a blob with wind with an ellipse-shaped fire front with maximum propagation speed $v_{\max}$, we can surround the blob with a teardrop-shaped fence built at rate $\rho > 2$.

Contents

1 Introduction

In recent years, there has been a significant increase in the frequency and intensity of forest fires [40]. This leads to an increase of carbon emissions, leading to even more wild fires. Therefore, it is of great importance that we know how to contain forest fires. To get a better understanding of containing the propagation of fire, we can use graphs. A graph is a pair of sets, the set of vertices and the set of edges, which connect vertices. Graphs can be used to model various spreading processes, such as rumour-spreading and epidemics. Perhaps of greater importance is finding control strategies for these spreading processes. This brings us to the firefighting problem on graphs. There is a wide range of questions concerning this problem. The topic of this bachelor thesis will be: firefighting on random geometric graphs. We have chosen these specific graphs, because they closely resemble actual forests. Just as in real forests, trees/vertices are placed randomly. Moreover, since fire can only spread to trees nearby, vertices are connected by an edge, when they are a certain distance apart. For containment strategies on these graphs, we follow the exposition in [3]. We build on [3] by introducing wind to the model.

1.1. Spreading processes

To introduce spreading processes on graphs we look at gossiping and broadcasting in communication networks [23]. In gossiping, everyone knows a unique piece of information and needs to communicate it to everyone else. By making calls people share all their information. The goal in the so-called Gossip Problem is to gain knowledge about the minimum number of calls needed such that everyone knows everything. This problem can be solved using communication graphs. Vertices are used for people and are connected by edges if people are allowed to call each other. Broadcasting is slightly different in the sense that only one person has some piece of information and needs to communicate it to everyone else. Similar to gossiping, the goal is to find the number of timesteps needed to complete broadcasting from one vertex/person. Each timestep, one call is made between two adjacent vertices.

We can also model the spread of a rumour as described in [15] and [18]. Rumour-spreading differs from broadcasting in the sense that, each timestep, all vertices that have received the rumour send it to a neighbour chosen uniformly at random, thus implying multiple calls per timestep. However, both are essentially the same, since they try to find the time it takes for the rumour to spread.

A topic of perhaps greater importance is the (computer) spread of viruses; we all remember the Covid epidemic a few years ago and a lot of us have seen movies in which computers take over the world. In [39] and [19] the spread of epidemics is modelled using graphs. Some initial set of vertices is infected. Infected vertices infect neighbours at some rate β. The difference with gossiping and broadcasting is that vertices can also recover at some rate δ. In [39] the model is used for computer viruses. They found an epidemic threshold for the ratio $\tau = \beta/\delta$, depending on the largest eigenvalue of a graph. Below the threshold the epidemic decays exponentially. To continue on the topic, [19] looked at the mean epidemic lifetime as a function of the number of vertices of various graphs.

1.2. Spreading processes with control strategies

Although it is very important to know how viruses or rumours spread, it is even better to know how to contain them. A teenager would give the world to prevent an embarrassing rumour from spreading and no-

body wishes for a Covid epidemic 2.0. In [33] an epidemic containment game is introduced. They build on the epidemic spreading model of [39] and [19] by independently giving vertices/people the choice to get secured/vaccinated. However, vaccination comes with a cost. Moreover, if people decide not to get vaccinated, this could slow down the ending of the epidemic. Therefore, there are also costs when the epidemic dies out slowly. The choices of each person give rise to a strategy profile. To look further into these strategy profiles, [33] looks at so-called Nash equilibria (NE). That is, a strategy profile where no player/node can benefit from switching his/her strategy, given that the others do not change their strategy. A different use of games on graphs is to prevent cyber attacks. [2] describes strategies for preventing worms from spreading by using alerts. Worms infect the vertices that can, in turn, be protected by worms. Another game is the game of cops and robbers as described in [6]. Imagine a robber is chased in your neighbourhood. You and your have decided to lock up your houses. Therefore the robber is restricted to run on the streets. We can imagine every intersection to be a vertex and every street to be an edge. We see the house the robber broke into as the starting vertex for the robber and the police station as the starting vertex of the cops. Lets say, we have a set of cops C and a robber R. Each turn the cops may move to their neighbouring vertices and then the robber may move to a neighbouring vertex. The goal of the cops is obviously to catch the robber and they do so by ending on the same vertex. The goal of the robber is to escape the cop. A winning strategy for the cops is a set of rules so that they catch the robber. A similar definition holds for the robbers. Of big importance is the cop-number of a graph, which is the smallest number of cops needed to catch the robber.

1.3. The firefighting problem

Closely related to the game of cops and robbers is this Bachelor thesis's subject: the Firefighting problem. The firefighting problem was first introduced by [22]. The firefighting problem is a discrete time process on graphs [37]. [16] describes it as follows. At time $t = 0$, fire breaks out on some set of vertices, giving a set of burning vertices. Each subsequent timestep, $f \in \mathbb{N}$ firefighters defend f vertices and the fire spreads to all undefended neighbours of burning vertices. Once on fire or defended, vertices remain in that state for the rest of the process. If the fire can no longer spread, the process is terminated and we have saved all protected and non-burning vertices.

We can consider multiple questions concerning the firefighter problem. We can strive to minimise the number of burned vertices or maximise the number of saved vertices. We can minimise the number of firefighters. We can answer these questions for particular graphs.

1.3.1. Surviving rate

In order to maximise the number of saved vertices,[28] introduced the surviving rate of a graph as the expected proportion of vertices that can be saved when a fire breaks out at a random vertex of the graph. There is a large amount of articles dedicated to finding the surviving rate of various graphs. In [28], they the surviving rate of trees. There are quite a few articles about outerplanar graphs: [28], [10] and [38]. In [25] and [14] planar graphs are studied. Moreover, in [11], they look at graphs with bounded tree width. In [31], it is shown that for any graph with average degree smaller than $30/11$, the surviving rate is bounded away from 0. This implies that for any such graph part of the vertices can be saved. Furthermore, in [26], the surviving rate of digraphs has also been studied [26].

1.3.2. Grid graphs

To study the question about minimising the number of firefighters, a lot of research has been dedicated to firefighting on grid graphs. You could see a grid graph as a tiling of shapes such as triangles, squares and hexagons. Grid graphs are infinite graphs and the goal in the firefighting problem is to contain the spread of the fire and in doing so, to use the least amount of fire fighters. In [30], they looked at firefighting on triangular grids. They used a strategy using 3 firefighters to contain the fire. They proved that, if we only start using firefighters at some time $t = k$, we can contain the fire at time $t = 18k + 3$, leaving at most $172k^2 + 58k + 5$ burned. In [20], it is conjectured that 2 firefighters are not enough to contain fire on triangular grids. In [20], the surviving rate of an infinite graph[1] is defined. They prove it is equal to $1/4$ for the infinite square grid. Intuitively this means that we can place firefighters such that 25% of the vertices is shielded from the fire. They provide a winning strategy for the (infinite) hexagonal grid using 1 firefighter each timestep, but using 2 additional firefighters at arbitrary timesteps. [12] improved on this result by providing a winning strategy using only 1 additional firefighter. They first contained the fire in a strip and then spiralled around

[1] We refer to the article itself for the definition.

it in order to contain it. It is conjectured by [20] that the fire cannot be contained without using additional firefighters. However, [29] prove that it is possible to contain the fire using no additional firefighters for an *oriented* hexagonal grid.

1.3.3. Decision problem

The firefighting problem has also been studied in the field of Computer Science. They used a decision problem in which the goal is to determine if there is a strategy in which a certain number of vertices can be saved. For example, [17] showed that this problem is NP-complete for trees of maximum degree 3, but in P for graphs of maximum degree 3 if the fire brakes out on a vertex of degree at most 2. Moreover, [24] showed that the problem is NP-complete for cubic graphs.

1.3.4. Variations

We have explored a wide range of research addressing questions about the regular firefighting problem. However, there is also research that modifies this problem to make it more realistic or more interesting. This research is dedicated to answer similar questions. One of these modified firefighting problems is studied in [7]. They studied the k-firefighting problem. Where the fire spreads to all unprotected neighbours in the regular problem, the fire chooses to spread to at most k unprotected neighbours in the k-firefighting problem. They define the k-surviving rate which is the same as the regular surviving rate with the new spreading condition. They bound the k-surviving rate of wheel graphs, prism graphs and random d-regular graphs. They also defined the surviving rate for infinite graphs and looked at infinite random graphs.

An other modified firefighting problem is the distance-restricted firefighting problem, as introduced in [9]. This is a more realistic approach to firefighters. The idea is that firefighters cannot move from every place to another as teleportation is not yet invented by our knowledge. Where we could place firefighters everywhere we want in the regular problem, this variant does not allow that. Firefighters can only move up to some fixed distance d. They look at the infinite square grid, the infinite strong grid and the infinite hexagonal grid.

1.4. Firefighting on random geometric graphs

This bachelor project is about firefighting on random geometric graphs. The motivation for this topic started when reading about firefighting on the hexagonal grid in [13]. They proved that the fire can be contained using 1 firefighter each turn, except for 1 turn where they used an extra firefighter. They drew the hexagonal grid on the Cartesian plane, allowing them to have vertices with coordinates without square roots or fractions. They defined sequences of coordinates of vertices that were protected. They used a strategy where they first contained the fire in a strip and then spiral around to contain the fire.

However, there are not many hexagonal-grid-shaped forests in the real world. The trees in a rainforest are not planted by humans and there position is therefore random. To make the problem more applicable to real life, we decided to look at random geometric graphs. The vertices of a random geometric graph are obtained by a homogeneous Poisson Point Process. Imagine λ tree seeds floating in the air and they all land randomly on a unit square of land. Then, imagine we have an infinite tiling of these squares of land and on each of these squares λ seeds land. Thereafter, we wait a few decades and we have our forest. We determine the number of seeds λ from a Poisson distribution of intensity lambda, giving us a homogeneous Poisson Point Process. We say that the fire can spread from one tree to the other if two trees are distance 1 apart. This gives us a random geometric graph in which vertices are connected when they are distance 1 apart. An example of a random geometric graph can be seen on the cover of this bachelor thesis. The image was taken from [34]. Where the firefighting problem on the hexagonal graph was deterministic, randomness is introduced for the random geometric graph. The goal is still to contain the fire using a minimum amount of firefighters. However, where a strategy either works or not for the hexagonal grid, a strategy works almost surely (with probability 1) for the random geometric graph. Most of this bachelor thesis follows the exposition in [3]. The technique used for the random geometric graph is very different from the technique with the hexagonal grid. While the problem for the hexagonal grid is solved directly from the graph, we translate the problem for the random geometric graph to the problem of surrounding a continuously growing blob with a fence. This blob starts as a unit disk and the boundary expands with rate 1 with time. The fence is then build at a certain building rate, corresponding to the number of firefighters. The goal is to minimise the fence building rate. This alternative approach is closely related to the dynamic blocking problem used in [8]. They use a set in the Euclidean plane expanding with time. They try to answer the question whether there is a strategy to block

the growth of the set by building barriers around it. Moreover, they look at optimal strategies, where they minimise the area burned by the fire and the cost of building the barrier (faster is more expensive).

1.5. Firefighting with wind

Although [3] does not reference any source related to the dynamic growth problem, we do think it is important to mention that there is a lot of research dedicated to what [3] named 'The blob'. Moreover, where [3] used a blob growing with unit rate in all directions, the dynamic blocking problem allows for a blob to grow differently in each direction. This bring us to the last part of this bachelor thesis: firefighting with wind. In the real world, wind plays a big role in firefighting. As explained in [1], the fire propagates differently in different directions under the influence of wind. In the direction of the wind the fire moves faster and in the direction opposite to the wind the fire moves slower. We try to translate the fence-building strategy of [3] to a fence-building strategy for a blob with wind.

1.6. Outline of the thesis

We start by introducing some basic definitions about graphs in Chapter 2. We define the homogeneous Poisson Point Process $\mathscr{P}_\lambda$ and the random geometric graph $G_{r,\lambda}$. In Chapter 3 we introduce the firefighting problem for graphs. We prove a theorem of [3] stating that the number of firefighters needed to contain the fire in a random geometric graph has an upper and a lower bound. Most of this thesis evolves around the upper bound. We make an analogy of firefighting on a random geometric graph with surrounding a blob growing continuously with unit rate with fences in Chapter 4. We use the method of [3] to find an upper bound for the fence building rate, which corresponds to an upper bound for the firefighters. We transfer the upper bound for the blob back to the upper bound for the firefighters in Chapter 5 following the exposition in [3]. We explain how [3] found a lower bound for the fence building rate in Chapter 6. We adjust the fence-building strategy for the blob to a strategy for a blob that has an ellipse-shaped fire front under the influence of wind in Chapter 7. We introduce a method of adjusting the fence-building strategy for the blob to a strategy for a general blob with wind. We end with a conclusion/discussion and some open problems in Chapter 8.

2

Random geometric graphs

We will define a graph and some properties of a graph in Section 2.1. In Section 2.2, we define the homogeneous Poisson point process, which is used to obtain the vertex set of a random geometric graph $G_{r,\lambda}$. In Section 2.3 we will give the full definition of $G_{r,\lambda}$, we will compute the degree of $G_{r,\lambda}$ and we will introduce the critical degree of $G_{r,\lambda}$ that determines when $G_{r,\lambda}$ has an infinite component.

2.1. Graphs

We will mostly follow the exposition in [4]. We give some important definitions about graphs.

Definition 2.1. A *graph* G is a pair of sets (V,E). V is known as the set of *vertices* and E as the set of *edges*. V is non-empty and E is a set of pairs of vertices, that is, $E \subseteq \{\{u,v\} : u,v \in V, u \neq v\}$.

Definition 2.2. A graph $G = (V,E)$ is called *infinite* if either the vertex set V is infinite or the edge set E is infinite.

Definition 2.3. We call two vertices $u,v \in V$ of a graph $G = (V,E)$ *adjacent* if they have an edge $\{u,v\} \in E$ between them.

Definition 2.4. Let $G = (V,E)$ be a graph and let $u \in V$ be a vertex. The set $N(u) = \{v \in V : \{u,v\} \in E\}$ is called the *neighbourhood* of the vertex u.

Definition 2.5. The *degree* of a vertex $u \in V$, denoted by $\deg(u)$, is the number of vertices in its neighbourhood.

Definition 2.6. A *path* from a vertex $u \in V$ to a vertex $v \in V$ is a collection of distinct vertices v_0, v_1,...,v_k, such that $u = v_0$, $v = v_k$ and $\{v_i, v_{i+1}\}$ are edges for all $0 \leq i \leq k-1$. There are no repeated vertices or edges.

Definition 2.7. A graph $G = (V,E)$ is *connected* if it has either one vertex $v \in V$ or there is a path between any two distinct vertices $v \in V$ and $u \in V$.

Definition 2.8. A *component* of a graph $G = (V,E)$ is a maximal connected subgraph $G = (V',E')$.

We give Example 2.1 to illustrate the definitions about graphs.

Example 2.1. In figure 2.1 we can see a graph to illustrate the definitions above. The circles are the vertices and they are connected by black lines, which are edges. Since there is a finite number of vertices and edges, it is a finite graph. The red and blue vertex are adjacent. The red vertex has 4 neighbours and therefore has degree 4. There is a path from the purple vertex to the orange vertex via the sky-blue vertices. The graph has two components: one on the left with 5 vertices and one on the right with 7 vertices

2.2. The homogeneous Poisson process

The vertex set of random geometric graphs is obtained by the homogeneous Poisson process. Therefore, we define the homogeneous Poisson process with intensity (density) λ. We will follow the exposition in [5].

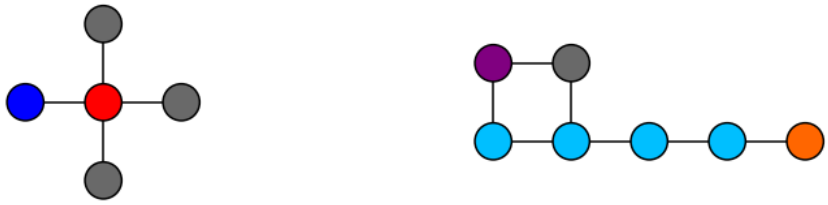


Figure 2.1: A graph with two components (Python, ChatGPT-assisted).

Definition 2.9. Let $\lambda > 0$ and let $\mathscr{P}_\lambda \subset \mathbb{R}^2$ be a random countably infinite set of points in the plane. We write $\mu_\lambda(U)$ for the number of points of $\mathscr{P}_\lambda$ in a bounded Borel set U. Note that $\mu_\lambda(U)$ is a random variable. We call $\mathscr{P}_\lambda$ a *homogeneous Poisson point process of intensity (density)* λ if the following two conditions hold:

i) If $U_1, \ldots, U_n$ are pairwise disjoint bounded Borel sets, then the random variables $\mu_\lambda(U_1), \ldots, \mu_\lambda(U_n)$ are independent;

ii) For every bounded Borel set U, the random variable $\mu_\lambda(U)$ is a Poisson random variable with mean $\lambda|U|$, where $|U|$ is the standard (Lebesgue) measure of U.

Remark 2.1. Note that the second condition of Definition 2.9 implies that for a Borel set Z of measure 0, the number of points in Z, $\mu_\lambda(Z)$, is a Poisson random variable with mean 0. Hence, the probability that $\mathscr{P}_\lambda$ has any point in Z is 0 and we can assume $Z \cap \mathscr{P}_\lambda = \emptyset$.

Example 2.2. As an example of a homogeneous Poisson point process, we can take independent identically distributed Poisson random variables $\{X_{i,j} : (i,j) \in \mathbb{Z}^2\}$, each with density $\lambda > 0$. Then as given in [21, p. 26], this $X_{i,j}$ takes values in $\{0,1,2,\ldots\}$ and its probability is given by

$$\mathbb{P}(X_{i,j} = k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad \text{for } k = 0,1,2,\ldots.$$

Thereafter, for each $X_{i,j}$ we create a unit square $\mathscr{Q}_{i,j}$, where we take $(i,j) \in \mathbb{Z}^2$ as the bottom-left corner.

$$\mathscr{Q}_{i,j} = \{(x,y) \in \mathbb{R}^2 : i \le x < i+1,\ j \le y < j+1\} \quad \text{for } (i,j) \in \mathbb{Z}^2.$$

For every $(i,j) \in \mathbb{Z}^2$, we select $X_{i,j}$ points independently and uniformly from $\mathscr{Q}_{i,j}$ and add these points to a set $\mathscr{Y}_{i,j}$. Then the union $\bigcup_{(i,j)\in\mathbb{Z}^2} \mathscr{Y}_{i,j}$ can be taken as a definition of $\mathscr{P}_\lambda$. We can see an example of the homogeneous Poisson point process with density 10 in Figure 2.2. We zoomed in to get a subset of $\mathbb{R}^2$ divided into unit squares. Each square $\mathscr{Q}_{i,j}$ contains $X_{i,j} X \sim \text{Poi}(10)$ points that are chosen uniformly.

2.3. The random geometric graph $G_{r,\lambda}$

We can now define a random geometric graph $G_{r,\lambda}$. We will follow the exposition in [5].

Definition 2.10. The *random geometric graph* $G_{r,\lambda}$ with real parameters $r > 0$ and $\lambda > 0$ has vertex set $\mathscr{P}_\lambda \subset \mathbb{R}^2$ obtained by a homogeneous Poisson point process (Definition 2.9). Two vertices are adjacent if they are at Euclidean distance at most r.

In Examples 2.3 and 2.4 we give two visualisations of the adjacency of two vertices in Definition 2.10.

Example 2.3. Let $G_{r,\lambda}$ be a random geometric graph with vertex set $V = \mathscr{P}_\lambda$. We draw closed disks $B_{i,r/2} \subset \mathbb{R}^2$ of radius $r/2$ around every vertex $v_i \in V$ with $i \in \mathbb{N}$. Observe that for any two distinct vertices v_i and v_j of which their disks overlap, the Euclidean distance between them is at most r, implying that the two vertices are adjacent. For any two distinct vertices v_i and v_j of which their disks do not overlap, the Euclidean distance between them is more than r and therefore v_i and v_j are not adjacent. In Figure 2.3 we show a close-up view of a random geometric graph $G_{1,2}$ as an example. Vertices are shown as red dots. They are surrounded by lavender disks of radius $1/2$ and connected by a black edge if the disks overlap.

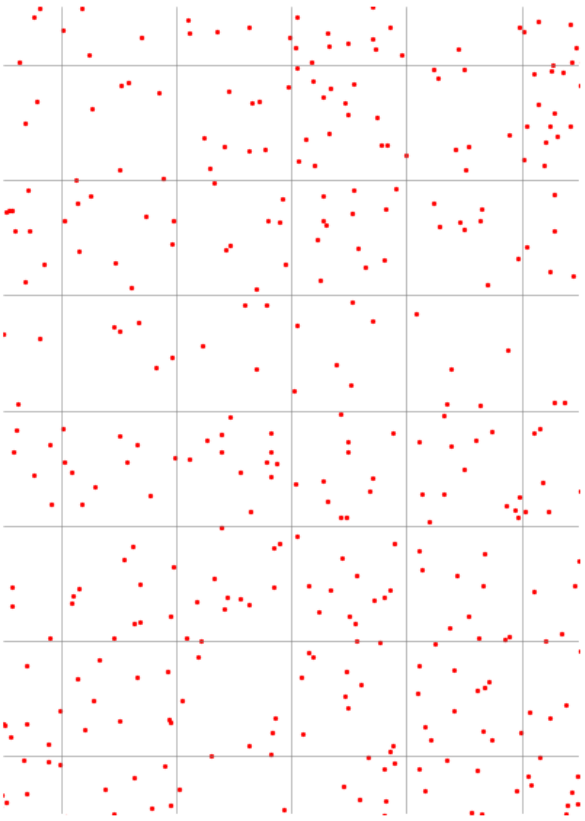


Figure 2.2: Homogeneous Poisson point process (Python, ChatGPT-assisted).

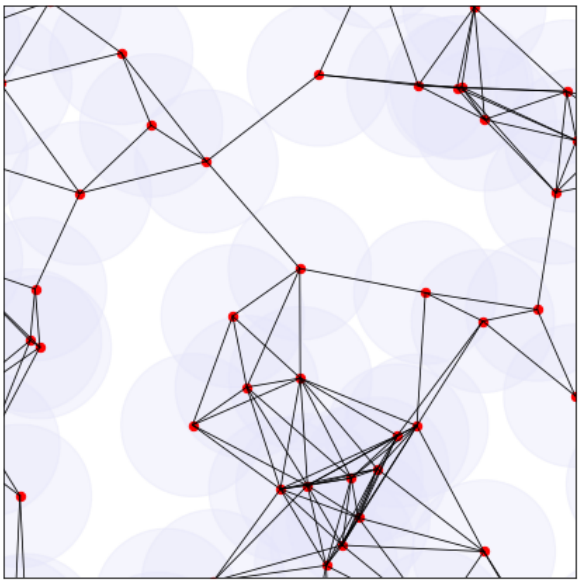


Figure 2.3: In a graph random geometric graph $G_{r,\lambda}$, two vertices are connected if the disks centred around them with radius $r/2$ overlap (Python, ChatGPT-assisted).

Example 2.4. Let $G_{r,\lambda}$ be a random geometric graph with vertex set $V = \mathscr{P}_\lambda$. We draw closed disks $B_{i,r} \subset \mathbb{R}^2$ of radius r around every vertex $v_i \in V$ with $i \in \mathbb{N}$. For a vertex $v_i \in V$, every vertex $v_j \in V$ that is inside the disk $B_{i,r}$ is at distance at most r from v_i and therefore adjacent to v_i. Any vertex v_j that is outside $B_{i,r}$ is at distance more than r away from v_i and therefore not adjacent to v_i. In Figure 2.4 we show a close-up view of a random geometric graph $G_{1,2}$ as an example. Vertices are shown as red dots and are connected by black

edges. We have added 1 lavender disk of radius 1 to keep the image clear. We can see that the vertex around which the disk is centred has 7 neighbours, which are all contained in the disk.

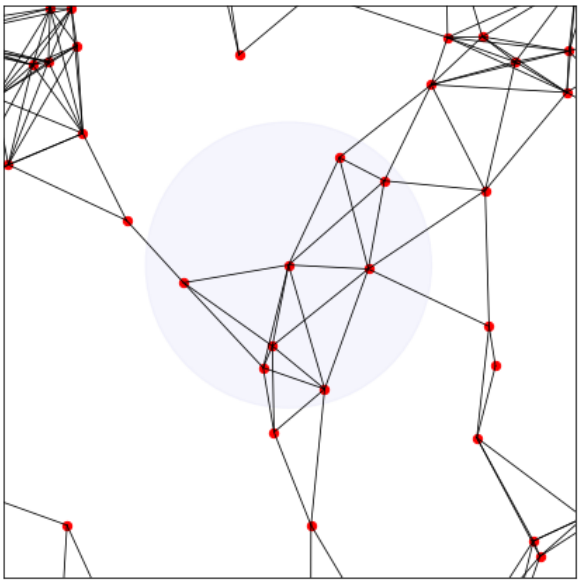


Figure 2.4: In a graph random geometric graph $G_{r,\lambda}$, a vertex is connected to all vertices that are contained in the disk of radius r around it (Python, ChatGPT-assisted).

Remark 2.2. Note that in Definition 2.10 the Euclidean distance between two vertices could also be enforced to be less than r in order for the two to be adjacent. This can be explained by using the interpretation of Example 2.4 and Remark 2.1. Since the boundary of a disk has measure 0, the probability that there is a vertex on the boundary of a disk is 0. Therefore, the probability that two vertices are exactly distance r apart is 0.

Example 2.4 helps to interpret the degree of a vertex u in $G_{r,\lambda}$.

Theorem 2.1. *The degree of a vertex in a random geometric graph $G_{r,\lambda}$ has the Poisson distribution with mean $\lambda\pi r^2$.*

Proof. Using Example 2.4 the neighbourhood of a vertex v_i of $G_{r,\lambda}$ is the set of vertices inside the closed disk $B_{i,r} \subset \mathbb{R}^2$. Since, by Definition 2.10, the vertices are obtained by a homogenous Poisson process of intensity λ. Therefore, by the second condition of Definition 2.9, the number of points in $B_{i,r}$ is Poisson distributed with mean $\lambda|B_{i,r}|$. Since $B_{i,r}$ is a closed disk of radius r we know that $|B_{i,r}| = \pi r^2$. Hence, the number of points in $B_{i,r}$ is Poisson distributed with mean $\lambda\pi r^2$. Since the degree of a vertex v_i in a random geometric graph is exactly the number of vertices in $B_{i,r}$, the degree of v_i has the Poisson distribution with mean $\lambda\pi r^2$. □

We can use the degree of vertices in a random geometric graph $G_{r,\lambda}$ to say something about the presence of an infinite component in the graph. This will be important in Chapter 3. Let $a = \lambda\pi r^2$ be the value of the degree in $G_{r,\lambda}$. Let $\theta(r,\lambda) = \theta(a)$ be the probability that the component of the origin is infinite. The component of the origin being finite corresponds to $\theta(a) = 0$. This only happens when a is sufficiently small. Furthermore, if a converges to ∞, then $\theta(a)$ converges to 1. We observe that, since $\theta(a)$ increases monotonically, there must be a critical degree a_c.

Definition 2.11. Let $G_{r,\lambda}$ be a random geometric graph. The degree of $G_{r,\lambda}$ is Poisson distributed with mean $a = \lambda\pi r^2$ as in Theorem 2.1. If $a < a_c$, with $a_c \in \mathbb{R}$, the probability that $G_{r,\lambda}$ has an infinite component centred at the origin is 0 and if $a > a_c$ this probability is strictly positive. The number a_c is the *critical degree* of $G_{r,\lambda}$.

Remark 2.3. We refer to [5] for further research on the critical degree.

3

The firefighting problem

This bachelor thesis is about finding a strategy for the firefighting problem on the random geometric graph $G_{1,\lambda}$ (Chapter 2). In Section 3.1 we define the firefighting problem for an arbitrary graph. In Section 3.2 we specify the firefighting problem for the random geometric graph $G_{1,\lambda}$. We follow the exposition in [3].

3.1. The Firefighting Problem

Let $G = (V, E)$ be a graph. The firefighting problem models the spread of a fire on a graph, along with an attempt to contain it using a limited number of firefighters. Initially, some vertices are on fire and, at each timestep, the fire spreads to unprotected neighbouring vertices. Subsequently, a limited number of vertices can be protected by firefighters to prevent the fire from spreading to them. The goal is to find a strategy for placing firefighters in order to contain the fire. We formally define the firefighting problem by Definition 3.1

Definition 3.1. Suppose we have a graph $G = (V, E)$. The firefighting problem models the spread of fire on the graph G, where we can protect vertices using firefighters. Let $F_t \subseteq V$ denote the set of vertices *on fire* at time t, and let $P_t \subseteq V$ be the set of *protected* vertices at time t. The process evolves in discrete time steps $t = 0, 1, 2, \ldots$. The *initial fire* at time $t = 0$ is the set $F_0 \subseteq V$, and a set $P_0 \subseteq V \setminus F_0$ of vertices is subsequently protected. At each time step $t \geq 1$, the fire spreads from burning vertices to adjacent unprotected vertices, and then a set of vertices is protected. The fire spreads to all unprotected neighbours of F_{t-1}.

$$F_t = F_{t-1} \cup A_t, \tag{3.1}$$

where

$$A_t = (V \setminus P_{t-1}) \cap N(F_{t-1}). \tag{3.2}$$

Next, up to f new vertices are protected:

$$B_t \subseteq V \setminus (F_t \cup P_{t-1}), \quad |B_t| \leq f. \tag{3.3}$$

Then

$$P_t = P_{t-1} \cup B_t. \tag{3.4}$$

We say that the fire is contained at time $t = t_{\text{end}}$ using f firefighters if $A_{t_{\text{end}}} = \emptyset$ and $|B_t| \leq f$ for every $t = 1, 2, \ldots, t_{\text{end}} - 1$.

3.2. Firefighting on $G_{1,\lambda}$

We can now define the firefighting problem for the random geometric graph $G_{1,\lambda}$. Since the set of vertices is no longer fixed, we define the number of firefighters sufficient to contain the fire on $G_{1,\lambda}$ as follows.

Definition 3.2. Let $f > 0$ and let $G_{1,\lambda}$ be a random geometric graph of density λ. If a fire starting at any finite set of vertices F_0 can be stopped by f firefighters with probability 1 (relative to the construction of $G_{1,\lambda}$), then f is said to be sufficient for λ. The infimum of all f sufficient for λ is denoted by f_λ.

We can couple the random geometric graphs G_λ for different intensities λ. Suppose we have a homogeneous Poisson point process of intensity $\lambda \geq \mu$, producing the vertex set V_λ. By independently retaining each vertex with probability μ/λ, we obtain a thinned vertex set $V_\mu \subseteq V_\lambda$ almost surely, which itself is a homogeneous Poisson point process of intensity μ. Therefore, there exists a coupling such that $G_\mu \preceq_{\text{st}} G_\lambda$. Subsequently, suppose that, by definition 3.2, f suffices for λ and we have any $\mu \leq \lambda$, then any $g \geq f$ suffices for μ. Therefore, f_λ is non-decreasing and we can find bounds for it. In chapter 2 we mention a critical degree $a = \lambda\pi r^2$. Since we now have the case $r = 1$, the critical degree equals $a = \lambda\pi$, which only depends on λ. Therefore we could also speak of a critical density λ_c. For $\lambda < \lambda_c$, the graph has only finite components, so the fire cannot spread across the whole graph and no firefighters are needed: $f_\lambda = 0$. Therefore, f_λ is not always bounded below by a positive multiple of λ. We will prove the upper bound of Theorem 3.1 in Chapters 4 and 5. We will briefly discuss the lower bound in Chapter 6.

Theorem 3.1. *There exist positive constants c_1, c_2, and c_3 such that*

$$c_2\lambda - c_3 \leq f_\lambda \leq c_1\lambda.$$

4

Surrounding a blob with a fence

To find an upper bound for f_λ (Chapter 3) in the random geometric graph $G_{1,\lambda}$ (Chapter 2), we can transfer the discrete firefighting problem, to the continuous problem of surrounding a blob with a fence. We will define the blob, the fence and the fence building rate in Section 4.1. In Section 4.2 we will substantiate how to go from the firefighting problem to the problem of surrounding a blob with a fence. Moreover, we introduce the concept that finding an upper bound for the building rate of the fence very much lightens the task of finding an upper bound for f_λ. Therefore, in Sections 4.3, 4.3 and 4.5, we will give bounds for the fence building rate.

4.1. Defining the blob and the fence

Definition 4.1. A *fence* at time $t \in \mathbb{R}_{\geq 0}$ is the set

$$F_t = \bigcup_{i=1}^{\infty} F_i(t), \tag{4.1}$$

where

$$F_i(t) = \begin{cases} \emptyset, & t < t_i^1, \\ \{\gamma_i(\tau) : \tau \in [t_i^1, t]\}, & t_i^1 \leq t \leq t_i^2, \\ \{\gamma_i(\tau) : \tau \in [t_i^1, t_i^2]\}, & t > t_i^2, \end{cases} \tag{4.2}$$

and each $\gamma_i : [t_i^1, t_i^2] \to \mathbb{R}^2$ is a continuously differentiable curve parametrized by time, with constant speed

$$\|\dot{\gamma}_i(\tau)\| = \rho_i > 0, \quad \forall \tau \in [t_i^1, t_i^2]. \tag{4.3}$$

Define the partition $P = \{t_0, t_1, \ldots, t_n\}$ of the interval $[0, t]$ by sorting all endpoints $\{t_i^1, t_i^2\}_{i=1}^{\infty} \cap [0, t]$ in increasing order. For each subinterval $I_k = [t_k, t_{k+1}]$ of the partition P, define the index set $S_k = \{i \in \mathbb{N} : I_k \cap [t_i^1, t_i^2] \neq \emptyset\}$. Then, for all k,

$$\sum_{i \in S_k} \rho_i = \rho, \tag{4.4}$$

where $\rho > 0$ is the *building rate* of the fence.

Definition 4.2. The *blob* is a path-connected region in $\mathbb{R}^2$ that expands continuously over time. The blob at time t is denoted by $B(t)$. Its growth may be obstructed by fences (Definition 4.1). At time $t = 0$, the blob is given by an arbitrary connected region $B(0) \subset \mathbb{R}^2$ that contains the origin. At any time $t \geq 0$, let $p \in \partial B(t)$ be a point on the boundary of the blob. For each angle $\theta \in [0, 2\pi]$, define the velocity vector at p in polar coordinates as:

$$\vec{v_p}(t, \theta) = \begin{cases} r(\theta) \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}, & \text{if } t < t_f(p, \theta), \\ \vec{0}, & \text{if } t \geq t_f(p, \theta), \end{cases} \tag{4.5}$$

where $r : [0, 2\pi] \to \mathbb{R}_{\geq 0}$ is a continuous function called the *growth rate* function. It specifies the speed at which the blob expands in each direction. The value $t_f(p, \theta)$ is the time at which the trajectory of p in direction θ intersects the fence $\gamma(t)$.

Remark 4.1. Since the blob $B(0)$ is path-connected and the growth process described is continuous over time without any splitting or disconnection, the blob $B(t)$ remains path-connected for all $t \geq 0$.

4.2. From firefighting on a graph to surrounding a blob with a fence

The main problem we need to consider when drawing an analogy between surrounding a blob with a fence and firefighting on a random geometric graph $G_{1,\lambda}$ (Definition 2.10) is that the first process is continuous and the second is discrete. Suppose we draw the graph $G_{1,\lambda}$ in the Euclidean plane. We remember from Definition 2.10 that two vertices are adjacent if they are at Euclidean distance smaller than 1. We assume that, at time $t = 0$, the fire starts out on all the vertices on a closed unit disk as can be seen in Figure 4.1. Without loss of generality, we can assume that this unit disk is centred around the origin. Moreover, we assume that we do not place any firefighters at any timestep. At time $t = 1$, the fire can spread to all vertices adjacent to the vertices initially on fire. Thus, the vertices on fire at time $t = 1$ are at distance at most 2 from the origin, implying that they are contained in a closed disk of radius 2 centred around the origin as can be seen in Figure 4.2. We can do the same for time $t = 2$ as can be seen in Figure 4.3. We can continue this process and conclude that at time $t = k$ the vertices on fire, when placing no firefighters, are contained in a closed disk of radius $k+1$ centred around the origin. We observe that this is exactly a blob starting as a unit disk centred around the origin and with growth rate function equal to 1 in all directions $\theta \in [0, 2\pi]$. Therefore, the spreading of fire and the growing of a blob have a clear resemblance when we choose a blob that starts as a unit disk centred at the origin and has unit growth rate in every direction. Note that at time $t = k$, the blob has radius $k+1$, assuming no fences are encountered up to time $t = k$. For the placement of firefighters, the resemblance to the fence is

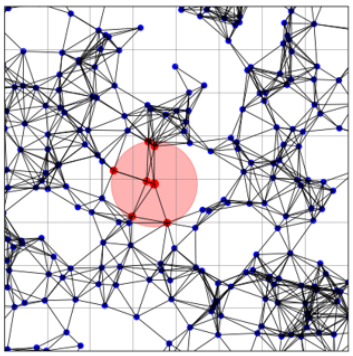


Figure 4.1: The graph and the blob at time $t = 0$ (Python, ChatGPT-assisted).

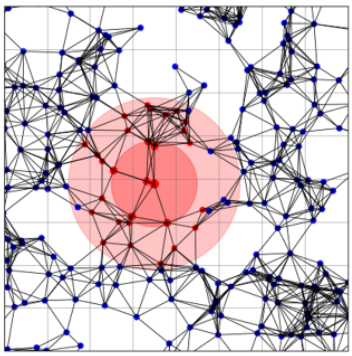


Figure 4.2: The graph and the blob at time $t = 1$ (Python, ChatGPT-assisted).

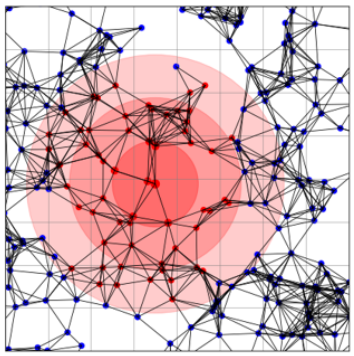


Figure 4.3: The graph and the blob at time $t = 2$ (Python, ChatGPT-assisted).

slightly more complicated. We will return to this similarity in Chapter 5; we argue that the fence building rate is proportional to the number of firefighters. Therefore, if we want to find an upper bound on the number of firefighters to be used to contain fire on a random geometric graph $G_{1,\lambda}$, we need to find an upper bound on the fence building rate ρ for the blob that grows at unit growth rate and starts as a unit disk. In Sections 4.3, 4.4 and 4.5 we will give three bounds for the fence building rate ρ following the exposition of [3].

4.3. Fence building strategy for $\rho > 2\pi$

We will prove Theorem 4.1.

Theorem 4.1. *The critical rate ρ_c for fence-building to surround the blob is at most 2π.*

Proof. To surround the blob with fence-building rate $\rho > 2\pi$ we will build a circular fence of radius d. An illustration of the circular fence can be seen in Figures 4.4, 4.5 and 4.6. We used $\rho = \sqrt{(2\pi)^2 + 0.1}$ so that the Matlab code does not run too long. To ensure $\rho > 2\pi$ we will take

$$\rho = \sqrt{(2\pi)^2 + \varepsilon} \quad \text{, for some } \varepsilon > 0. \tag{4.6}$$

We let time $t = s$ be the time at which the fence is finished and simultaneously hit by the edge of the blob. Since the radius of our blob at time t is given by the function $r(t) = t+1$, we know that at time $t = s$, $r(s) = s+1$, giving us $d = s+1$. Moreover, we know that the perimeter of the fence is $2\pi d$. Furthermore, the fence was built at rate ρ, implying that, at time $t = s$ the fence has length $s\rho$. Therefore, we have

$$2\pi(s+1) = s\rho \tag{4.7}$$

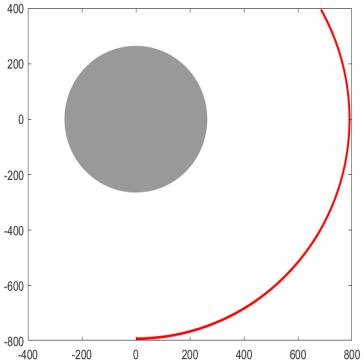


Figure 4.4: The blob and the circular fence at time one third of the way (Matlab).

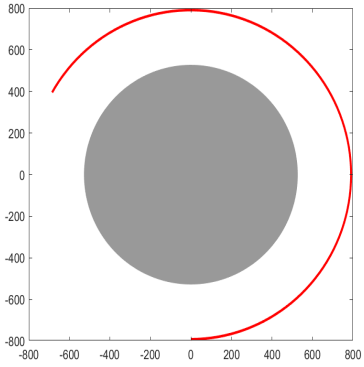


Figure 4.5: The blob and the circular fence at time two thirds of the way (Matlab).

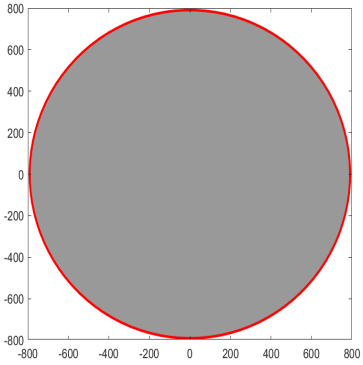


Figure 4.6: The blob is surrounded by a circular fence (Matlab).

where we used $d = s + 1$. Solving for s yields

$$s = \frac{2\pi}{\rho - 2\pi}. \tag{4.8}$$

Using $d = s + 1$, we get

$$d = \frac{\rho}{\rho - 2\pi}. \tag{4.9}$$

We can conclude that, if we build a circular fence of radius $d = \rho/(\rho - 2\pi)$, we have contained the blob at time $s = 2\pi/(\rho - 2\pi)$. This finishes the proof. □

4.4. Fence building strategy for $\rho > 4$

We will prove Theorem 4.2.

Theorem 4.2. *The critical rate ρ_c for fence-building to surround the blob is at most* 4.

Proof. To surround the blob with fence-building rate $\rho > 4$ we will build a rectangular fence. To ensure $\rho > 4$ we will take

$$\rho = \sqrt{16 + \varepsilon} \quad \text{, for some } \varepsilon > 0. \tag{4.10}$$

We start by building 4 vertical fences up and down from $(-d, 0)$ and $(d, 0)$, each at rate $\rho/4$, ensuring total building rate ρ. An illustration of the process can be seen in Figures 4.7, 4.8 and 4.9. We used $\rho = \sqrt{4^2 + 0.7}$ so that the Matlab code does not run too long.

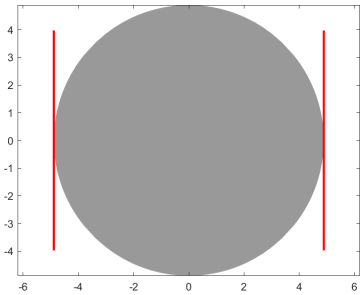


Figure 4.7: The blob hits the vertical fences (Matlab).

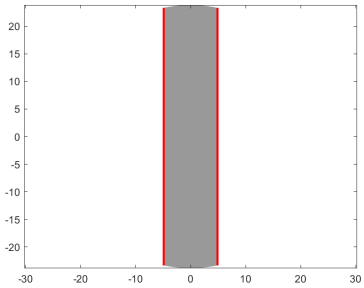


Figure 4.8: The blob and fences at time $t = s$ (Matlab).

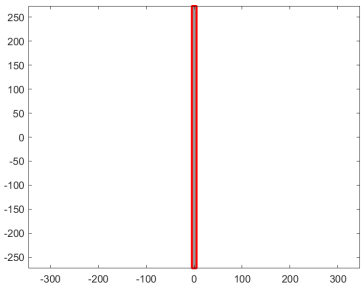


Figure 4.9: The blob is surrounded by a rectangular fence (Matlab).

We will choose d minimal such that the blob cannot spill over the vertical fences. The goal is to build these vertical fences until it is possible to close of the blob with horizontal fences into a rectangular shaped fence. By similarity, we only show the calculations for the fence going up from $(d, 0)$. As the blob starts as a unit disk centred around the origin $(0, 0)$ and it has unit growth rate in all directions, the blob reaches the fence at time $t = d - 1$. For time $t \geq d - 1$, the blobs edge will advance along the fence and thus intersect with the half-line

$\{(d, y) : y \geq 0\}$ at (d, y). Since the blob has radius $t+1$ at time $t \geq d-1$, we can use the Pythagorean theorem to get $y^2 + d^2 = (t+1)^2$. Rewriting gives us the y-coordinate of the edge of the blob along the fence.

$$y_{\text{blob}}(t) = \sqrt{(t+1)^2 - d^2} \tag{4.11}$$

We also know the y-coordinate of the fence.

$$y_{\text{fence}}(t) = \frac{\rho t}{4} \tag{4.12}$$

Since we do not want the blob to cross the vertical fence and we want to be able to close of the blob with a vertical fence, the vertical fence should be ahead of the edge of the blob when we start building the horizontal fence. Therefore, we need $y_{\text{fence}}(t) \geq y_{\text{blob}}(t)$ for all $t \geq 0$. Since $y_{\text{blob}}(t), y_{\text{fence}}(t) \geq 0$ for all $t \geq 0$, we can square both sides of the inequality and thus we need $(\rho/4)^2 t^2 - (t+1)^2 + d^2 \geq 0$. This means that we should have a non-negative minimum of the function

$$f(t) = \left(\frac{\rho}{4}\right)^2 t^2 - (t+1)^2 + d^2. \tag{4.13}$$

To find the minimum of $f(t)$, we set the derivative of $f(t)$ with respect to t to 0 and solve for t, yielding $t = s$, with

$$s = \frac{16}{\rho^2 - 16}. \tag{4.14}$$

To check that it is indeed a minimum, we compute the second derivative with respect to t to be $2(\rho/4)^2 - 2 > 0$, since $\rho > 4$. Subsequently, we can set $f(s) \geq 0$ and solve for d, giving $d^2 \geq (s+1)^2 - (s\rho/4)^2 \geq 0$. Since it can be checked that $d^2 \geq 1$, which implies $d \geq 1$, minimising d is the same as minimising d^2 and thus we get

$$d = \sqrt{(s+1)^2 - (s\rho/4)^2}. \tag{4.15}$$

Because our vertical fences are all a horizontal distance d apart from the vertical line through the origin, we can build 4 horizontal fences of length d at rate $\rho/4$, starting at the endpoints of the vertical fences. Again, by similarity, we only show calculations for the horizontal fence corresponding to the vertical fence going up from $(0, d)$. Let t_{switch} be the time at which we stop building the vertical fence and start building the horizontal fence. Let t_{hor} be the time it takes to finish the (horizontal) fence. The horizontal will have length d when finished. Since we build it at rate $\rho/4$ it will have length $t_{\text{hor}}\rho/4$ and thus we must have $d = t_{\text{hor}}\rho/4$, giving $t_{\text{hor}} = 4d/\rho$. The vertical fence will have length $t_{\text{switch}}\rho/4$ by the time it is finished. Since we stop building the vertical fence after time $t = t_{\text{switch}}$, it will still have length $t_{\text{switch}}\rho/4$ when we finish our total fence, call it time t_{end}. Note that $t_{\text{end}} = t_{\text{switch}} + t_{\text{hor}}$. Since we want the blob to be contained inside the rectangle of fences. By the time we finish the horizontal fence, the most northern point of the blob must be at the same height as the vertical fence. We finish the horizontal fence at time $t = t_{\text{end}}$, so the top of the blob will be at $y = t_{\text{end}} + 1$ and the fence at height $y = t_{\text{switch}}\rho/4$. Using $t_{\text{end}} = t_{\text{switch}} + t_{\text{hor}}$, this implies

$$t_{\text{switch}} + \frac{4}{\rho}d + 1 = \frac{\rho}{4}t_{\text{switch}}. \tag{4.16}$$

and solving the equation for t_{switch} we get

$$t_{\text{switch}} = \frac{16d + 4\rho}{\rho^2 - 4\rho}. \tag{4.17}$$

Therefore, if we build 4 vertical fences going up and down from $(-d, 0)$ and $(d, 0)$ until time $t = t_{\text{switch}}$ and then build 4 horizontal fences starting from the endpoints of the vertical fences until time $t = t_{\text{switch}} + 4d/\rho$, we can contain the blob with a rectangular fence built at rate $\rho > 4$. This finishes the proof. □

4.5. Fence building strategy for $\rho > 2$

We will prove Theorem 4.3.

Theorem 4.3. *The critical rate ρ_c for fence-building to surround the blob is at most* 2.

Proof. To surround the blob with fence-building rate $\rho > 2$ we will build a teardrop-shaped fence. To ensure $\rho > 2$ we will take

$$\rho = \sqrt{4+\varepsilon} \quad \text{, for some } \varepsilon > 0. \tag{4.18}$$

Moreover, we define

$$s = \frac{4}{\rho^2 - 4} = \frac{4}{\varepsilon} \tag{4.19}$$

where we used (4.18) to simplify. The blob starts as a unit disk centred around the origin $(0,0)$. We will start by building a horizontal fence at at the point $(0,-d)$, which is distance $d-1$ south of the origin. The formula for d is

$$d = \sqrt{(s+1)^2 - \frac{s^2\rho^2}{4}} = \frac{\rho}{\sqrt{\varepsilon}}. \tag{4.20}$$

where we use (4.18) for ρ and (4.19) for s. We have chosen (4.19) and (4.20) in a clever way that will become clear later in the proof. From the point $(0,-d)$ we will built a horizontal fence to the west at rate $\rho/2$ and to the east at rate $\rho/2$, giving us the assumed total rate of ρ. Since the west and east side of the fence have similar characteristics, we will only show calculations for the east side of the fence. At time $t = d-1$, the blob will intersect with the fence at $(0,-d)$, as can be seen in Figure 4.10. We use $\rho = \sqrt{2^2+0.7}$ for images if the blob with teardrop-shaped fence so that the Matlab code does not run too long.

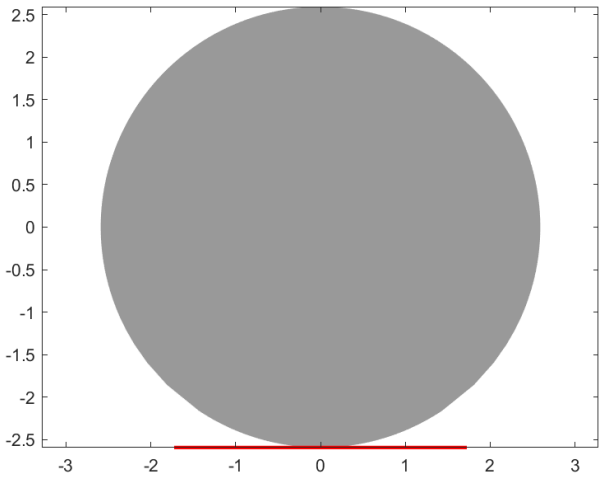


Figure 4.10: The blob hits the horizontal fence at time $t = d-1$ (Matlab).

Then, for time $t \geq d-1$, the blobs edge will advance along the fence and thus intersect with the half-line $\{(x,-d) : x \geq 0\}$ at $(x,-d)$, as can be seen in Figure 4.11.

Since the blob has unit growth rate and starts as a unit disk centred at the origin, it will have radius $t+1$ at time $t \geq d-1$. Moreover, it will intersect the fence at point $(x,-d)$, which is at distance $\sqrt{x^2+d^2}$ from the origin. Combining the two gives us (4.21).

$$\sqrt{x^2+d^2} = t+1. \tag{4.21}$$

We can rearrange (4.21) to get (4.22) for the x-coordinate of the edge of the blob, along the fence, over time. Moreover, by using the unit growth rate of the fence we can get (4.23) for the x-coordinate of the fence over time. Giving us (4.22) and (4.23).

$$x_{\text{blob}}(t) = \sqrt{(t+1)^2 - d^2} \tag{4.22}$$

$$x_{\text{fence}}(t) = \frac{t\rho}{2}. \tag{4.23}$$

We can plot (4.22) and (4.23) against time $t \geq d-1$ as can be seen in Figure 4.13. We see that at the time the blobs edge hits the fence, time $t = d-1$, the fence is ahead of the blob. Then, at time $t = s$, which we specifically chose, the blob catches up with the fence, but for $t > s$, the fence gets ahead again. An illustration of the situation at time $t = s$ can be seen in Figure 4.12. We can show the catching up of the blob with the fence at time $t = s$ algebraically. Since we grow the east part of the fence at rate $\rho/2$, it will reach from the

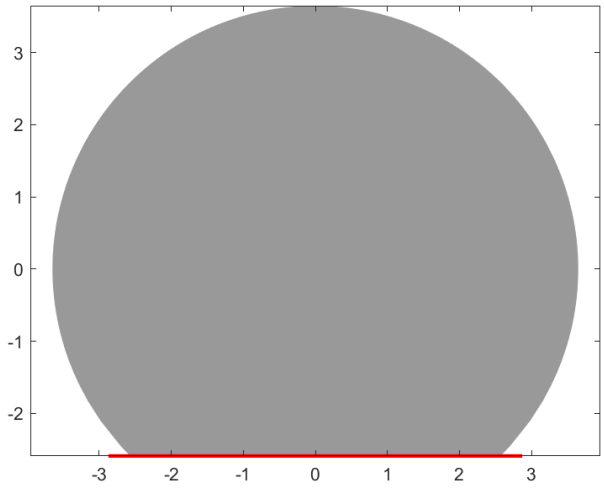


Figure 4.11: The blob's edge grows along the horizontal fence (Matlab).

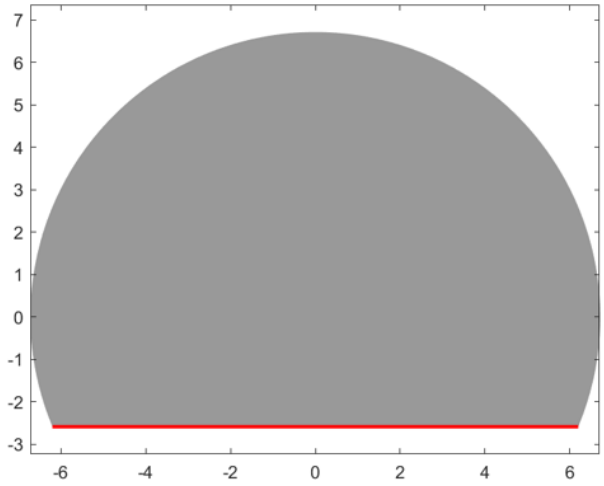


Figure 4.12: The blob's edge meets the horizontal fence (Matlab).

origin $(0,0)$ to the point $(s\rho/2,-d)$ at time $t=s$. For this point $(s\rho/2,-d)$ we can calculate its distance to the origin to be $\sqrt{(s\rho/2)^2+(-d)^2}$, which simplifies to $s+1$. Since the blob has unit growth rate and starts as a unit disk centred at the origin, it will have radius $s+1$ and therefore, the blob and the fence intersect at time $t=s$. Moreover, we can interpret Figure 4.13 in terms of velocity. To catch up with the fence, the blob needs to have a higher growth-velocity than the fence. However, after the blob catches up with the fence, it gets behind again, implying a higher growth-velocity of the fence. This would suggest that, at time $t=s$, the growth-velocity of the fence and the blob are equal. We can show this algebraically. By differentiating with respect to t and rearranging (4.21), we can find the horizontal advance rate of the edge of the blob along the half-line $\{(x,-d) : x \geq 0\}$ at $(x,-d)$ to be

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t}(t) &= \frac{t+1}{x} \\ \frac{\mathrm{d}x}{\mathrm{d}t}(s) &= \frac{s+1}{\left(\frac{s\rho}{2}\right)} \end{aligned} \tag{4.24}$$

where the second line represents the situation at time $t=s$, when the blob is at $(-s\rho/2,-d)$.If we simplify (4.24), we get a horizontal advance rate of the blob of $\rho/2$, which is exactly equal to the rate of growth of the fence. Therefore, at time $t=s$, the growth-velocity of the fence and the blob are equal. As we could see in Figure 4.13, the blobs edge gets behind of the fence again, implying that the fence grows faster than the blob. Therefore, in order to keep up with the blob, we do not have to build a horizontal fence anymore. The fence

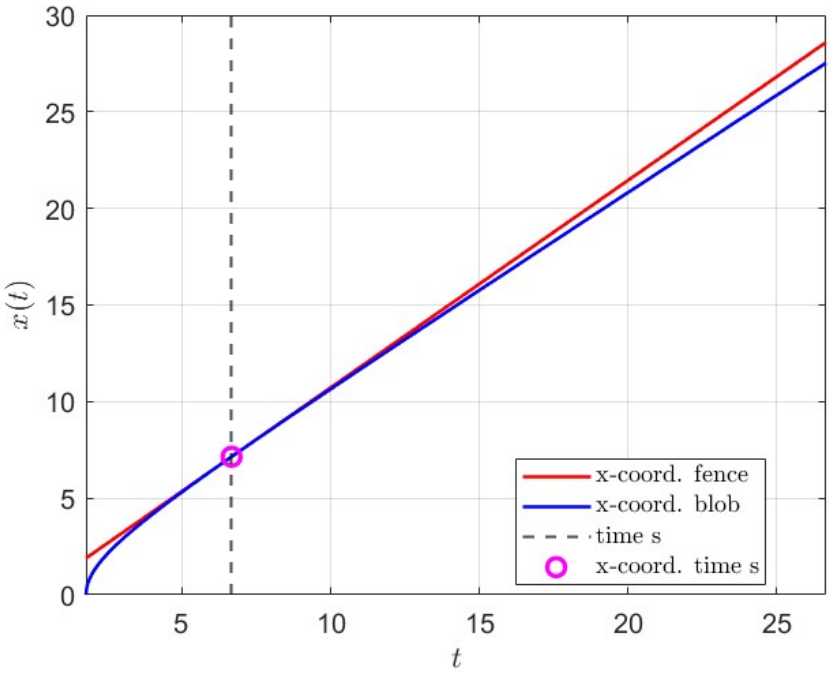


Figure 4.13: We see a plot of the x-coordinate of the fence (in red) and the x-coordinate of the blob along the half-line $\{(x,-d) : x \geq 0\}$ (in blue). The black vertical line represents the time s of which we also see a circle around the corresponding x-coordinate of the fence and the blob along the half-line. We used $\varepsilon = 0.7$ (Matlab).

can follow the blob upwards. Thus, intuitively, we 'bend' the fence upwards and grow it along the edge of the blob at rate $\rho/2$, to get our desired teardrop-shaped fence. An illustration of how the fence is bent can be seen in Figures 4.14 and 4.15. For $t \geq s$, we will express the growth points in polar coordinates $(r(t), -\theta(t))$

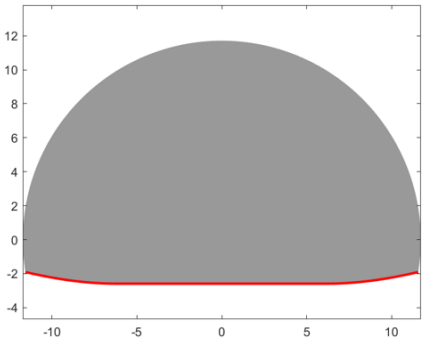


Figure 4.14: The fence is bend upwards along the blob (Matlab).

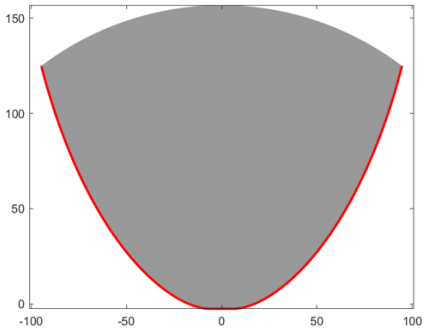


Figure 4.15: The fence is bend further upwards along the blob (Matlab).

for the west side and $(r(t),\theta(t))$ for the east side of the fence. We let $\theta = 0$ represent the south. As before, we continue our calculations for the east side. We can interpret the growth points as moving points in the plane and express them as vector-valued functions over time. We can write the position vector $\vec{x}(t)$ in Cartesian coordinates

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix}. \tag{4.25}$$

In polar coordinates this gives

$$\vec{x}(t) = \begin{bmatrix} r(t)\cos(\theta(t)) \\ r(t)\sin(\theta(t)) \end{bmatrix}. \tag{4.26}$$

To compute the velocity vector $\vec{v}(t)$, we take the derivative with respect to time of the position vector.

$$\begin{aligned}\vec{v}(t) &= \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} r(t)\cos(\theta(t)) \\ r(t)\sin(\theta(t)) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\mathrm{d}r}{\mathrm{d}t}(t)\cos(\theta(t)) - r(t)\sin(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) \\ \frac{\mathrm{d}r}{\mathrm{d}t}(t)\sin(\theta(t)) + r(t)\cos(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) \end{bmatrix}\end{aligned} \tag{4.27}$$

To compute the arc-speed of the growth points we take the Euclidean norm of the velocity vector $\vec{v}(t)$ [35]. Since we build the fence along the blob, we know $r(t) = t+1$, so $\frac{\mathrm{d}r}{\mathrm{d}t}(t) = 1$.

$$\begin{aligned}\|\vec{v}(t)\| &= \left\| \begin{bmatrix} \frac{\mathrm{d}r}{\mathrm{d}t}(t)\cos(\theta(t)) - r(t)\sin(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) \\ \frac{\mathrm{d}r}{\mathrm{d}t}(t)\sin(\theta(t)) + r(t)\cos(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) \end{bmatrix} \right\| \\ &= \sqrt{\left(r(t)\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)\right)^2 + \left(\frac{\mathrm{d}r}{\mathrm{d}t}(t)\right)^2} \\ &= \sqrt{(t+1)^2\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)\right)^2 + 1}\end{aligned} \tag{4.28}$$

Since we grow the fence at rate $\rho/2$, we have an arc-speed of $\rho/2$, giving us (4.29).

$$\frac{\rho}{2} = \sqrt{(t+1)^2\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)\right)^2 + 1} \tag{4.29}$$

Note that $\theta(t)$ is the only unknown in (4.29). We can get an expression for $\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)$ by rewriting equation 4.29.

$$\begin{aligned}\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) &= \frac{\sqrt{\frac{\rho^2}{4} - 1}}{t+1} \\ &= \frac{\sqrt{\varepsilon}}{2(t+1)}\end{aligned} \tag{4.30}$$

where we used $\rho = \sqrt{4+\varepsilon}$, implying $\rho^2/4 - 1 = \varepsilon$, for the second line. Integrating with respect to t gives us the desired formula for $\theta(t)$:

$$\theta(t) = C + \frac{\sqrt{\varepsilon}}{2}\log(t+1), \quad C \in \mathbb{R}. \tag{4.31}$$

To find the constant $C \in \mathbb{R}$, we have to know one value of $\theta(t)$, for a certain time t. Fortunately, we know enough about the situation at time $t = s$ to find $\theta(s)$. We take a right triangle with an angle $\theta(s)$. The hypotenuse side of the triangle is the radius of the blob and the opposite side is the horizontal part of the fence. Note that the sine of the angle $\theta(s)$ is equal to the length of the opposite side divided by the length of the hypotenuse side. We know that, at time $t = s$, the radius of the blob has length $r(s) = s+1$. Moreover, the fence is build from $(0,-d)$ to $\left(s\rho/2,-d\right)$ giving us the length of the opposite side to be equal to $\sqrt{\left(s\rho/2-0\right)^2 + (-d+d)^2}$. Therefore, for the sine of the angle $\theta(s)$, we have

$$\sin(\theta(s)) = \frac{\sqrt{\left(\frac{s\rho}{2} - 0\right)^2 + (-d+d)^2}}{s+1} = \frac{\frac{s\rho}{2}}{s+1} = \frac{2}{\rho}. \tag{4.32}$$

By taking the inverse sine, we get

$$\theta(s) = \sin^{-1}\left(\frac{2}{\rho}\right). \tag{4.33}$$

We can know calculate the constant $C \in \mathbb{R}$ in formula 4.31.

$$C = \sin^{-1}\left(\frac{2}{\rho}\right) - \frac{\sqrt{\varepsilon}}{2}\log(s+1) = \sin^{-1}\left(\frac{2}{\rho}\right) - \frac{\sqrt{\varepsilon}}{2}\log\left(\frac{\rho^2}{\varepsilon}\right). \tag{4.34}$$

By using the expression for C in the formula for $\theta(s)$, we get

$$\begin{aligned}\theta(t) &= \sin^{-1}\left(\frac{2}{\rho}\right) + \frac{\sqrt{\varepsilon}}{2}\log\left(\frac{\varepsilon(t+1)}{\rho^2}\right) \\ &= \sin^{-1}\left(\frac{2}{\rho}\right) + \frac{\sqrt{\varepsilon}}{2}\log\left(\frac{\varepsilon(t+1)}{\varepsilon+4}\right)\end{aligned} \tag{4.35}$$

where we used (4.18) for ρ. Since the teardrop is finished north of the origin, the growth point of the east side of the fence collides with that of the west side of the fence when $\theta = \pi$, giving the teardrop-shaped fence as can be seen in Figure 4.16. If we say the collision happened at time $t = t_{\text{end}}$, then $\theta(t_{\text{end}}) = \pi$ and we can use

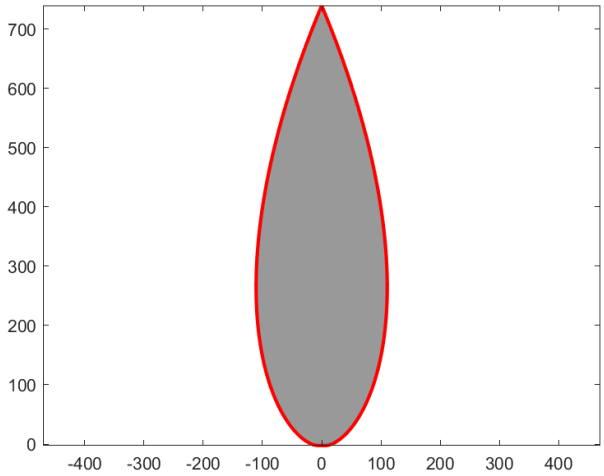


Figure 4.16: The blob's edge meets the horizontal fence (Matlab).

(4.35) to get an implicit expression for t_{end}.

$$\pi = \sin^{-1}\left(\frac{2}{\rho}\right) + \frac{\sqrt{\varepsilon}}{2}\log\left(\frac{\varepsilon(t_{\text{end}}+1)}{\varepsilon+4}\right) \tag{4.36}$$

Rewriting formula 4.36 gives the explicit expression for t_{end}.

$$t_{\text{end}} = \frac{\varepsilon+4}{\varepsilon}\exp\left(\frac{2}{\sqrt{\varepsilon}}\left(\pi - \sin^{-1}\left(\frac{2}{\rho}\right)\right)\right) - 1 \tag{4.37}$$

This finishes the proof. □

Remark 4.2. Note that (4.37) implies that the time it takes to finish the teardrop is exponential in $2/\sqrt{\varepsilon}$. To argue that the length of the teardrop is also exponential in $2/\sqrt{\varepsilon}$, we need to compute the arc-length. Our teardrop consists of two horizontal parts and two curved parts. However, we always build the fence at building rate $\rho/2$, so there is no difference between the horizontal and vertical part of the fence. Hence, we take the integral from $t = 0$ to $t = t_{\text{end}}$ of $\rho/2$ for the west part and for the east part of the fence. It follows that the arc-length is equal to

$$\begin{aligned}2\int_0^{t_{\text{end}}} \frac{\rho}{2}\,dt &= 2\left[\frac{\rho}{2}t\right]_0^{t_{\text{end}}} \\ &= \rho\, t_{\text{end}} \\ &= 4\sqrt{\frac{4}{\varepsilon^2} + \frac{1}{\varepsilon}}\exp\left(\frac{2}{\sqrt{\varepsilon}}\left(\pi - \sin^{-1}\left(\frac{2}{\rho}\right)\right)\right)\end{aligned} \tag{4.38}$$

where we used (4.37) for t_{end} and (4.18) for ρ. We observe that (4.38) implies that the length of the teardrop is also exponential in $2/\sqrt{\varepsilon}$.

Remark 4.3. We can calculate the angle of the fence with the vertical. We take a right triangle, with the angle of the fence as one of the not-right angles and the fence building rate tangent to the fence as the hypotenuse side and the vertical fence building rate as the adjacent side. We know that the fence building rate tangent to the fence is equal to $\rho/2$. We also know that the vertical fence building rate is equal to 1, because at the north of the origin the blob grows upward at rate 1 and the fence follows the edge of the blob. Therefore, the angle of the fence with the vertical is given by (4.39).

$$\cos^{-1}\left(\frac{2}{\rho}\right) \tag{4.39}$$

To conclude, as ε converges to 0, ρ converges to 2 and therefore $\cos^{-1}\left(2/\rho\right)$ converges to $\cos^{-1}(1)$, which is equal to 0 giving our teardrop an infinitely pointy shape.

5

Upper bound for random geometric graph

In Chapter 4, we compared firefighting on a random geometric graph with building a fence around a blob. In this chapter, we will use the strategy for the blob to find an upper bound for the number of firefighters needed to contain a fire on a random geometric graph. We will prove Theorem 5.1 following the exposition in [3].

Theorem 5.1. *The number of firefighters, f_λ, needed to contain the fire, is almost surely less than 2λ for all $\lambda > 0$.*

Proof. If we want to contain the fire we have to make a barrier of firefighters that surrounds the fire. Two vertices in our random geometric graph are adjacent if the Euclidean distance between the two is smaller than 1. Therefore, our barrier will have to be of width 1 so that the fire cannot jump across the barrier. An illustration can be seen in Figure 5.1. The red vertices are the vertices on fire. The yellow vertices are the protected vertices inside the barrier that has lavender colour. The saved vertices are coloured gray. We can see that the fire cannot cross the barrier and spread to the gray vertices, because all the vertices inside the barrier of width one are protected. Our barrier will be the region R between two curves C_1 and C_2. We will

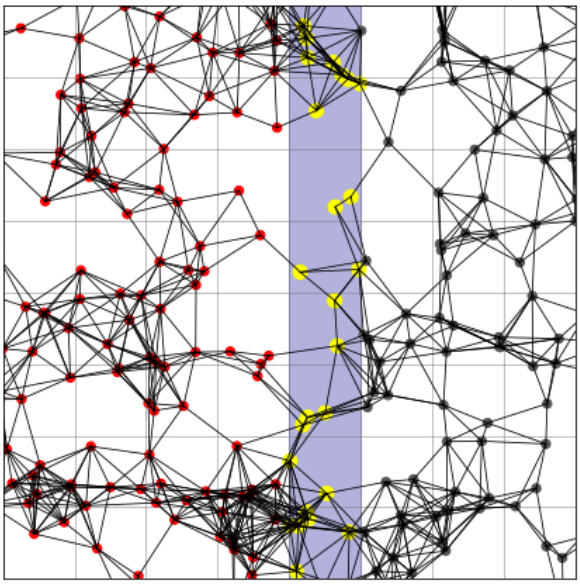


Figure 5.1: The barrier needs to be of width 1 (Python, ChatGPT-assisted).

contain the fire in the interior of C_1, which, in turn, will be contained in the interior of C_2. The curves will be distance 1 apart from each other, ensuring width 1 for region R. As with the blob, our goal is to make a teardrop-shaped barrier. We will take the outside curve C_2 as our fence in the blob-strategy and we will take $\rho = 2(1+\delta)$, for $\delta > 0$, as our building rate. Note that, similarly to the teardrop-strategy, $\rho > 2$. Our initial blob

will be a disk with radius r_1 centred around the origin. To ensure the fire never crosses our protection region R, we assume the initial fire is contained inside a unit disk of radius $r_2 = r_1 - 2$. For the firefighting problem, our fence C_2 contains λ firefighters per unit length. Therefore, we will need more than 2λ firefighters per unit length of C_2. We will take our number of firefighters to be $f = 2\lambda(1+\delta)^2$. As with the blob, we will consider an east and a west part of our barrier. For the east part we take C_1^{east} as our inside curve and C_2^{east} as our outside curve. Similarly, we take C_1^{west} and C_2^{west} for the west part. Because of similarity, we will only show the steps for the east side from now on. Each curve is naturally parametrised by time t. Since we build the outer curve, C_2^{east}, at rate $\rho/2$, we get that the arc-length of $\{C_2^{\text{east}}(t) : t_1 \le t \le t_2\}$ is $(t_2 - t_1)\rho/2$. If we look at the set $\mathbb{R}^2 \setminus C_2$, we see a bounded region, enclosed by C_2. We define C_1 as the set of all points in this bounded region that are at distance 1 from C_2. Since C_2 is convex, each point $C_2^{\text{east}}(t)$, will be at distance 1 from a unique point of C_1^{east}, which we will call $C_1^{\text{east}}(t)$. Since C_1^{east} is the inner curve of the region R, we get that $\{C_1^{\text{east}}(t) : t_1 \le t \le t_2\}$, has arc-length at most $(t_2 - t_1)\rho/2$. We can now take a look at a small piece of our barrier R for the time-interval $[t_1, t_2]$, call it $R^{\text{east}}[t_1, t_2]$. This is the region enclosed by the curves $\{C_1^{\text{east}}(t) : t_1 \le t \le t_2\}$ and $\{C_2^{\text{east}}(t) : t_1 \le t \le t_2\}$ and the unit line segments from $C_1^{\text{east}}(t_1)$ to $C_2^{\text{east}}(t_1)$ and from $C_1^{\text{east}}(t_2)$ to $C_2^{\text{east}}(t_2)$. This region has area at most $(t_2 - t_1)\rho/2$. If we do the same for the west side of the barrier we get that the area of the region $R^{\text{west}}[t_1, t_2]$ is at most $(t_2 - t_1)\rho/2$. We start building our barrier at time $t_1 = 0$. Suppose we are at time $t_2 = t$, then our total barrier is $R_t = R^{\text{west}}[0, t] \cup R^{\text{east}}[0, t]$, which has area at most $2(t-0)\rho/2 = t\rho$. Since in our random geometric graph the number of vertices per unit area is Poisson distributed with densityλ, the number of trees in R_t, N_t, is Poisson distributed with density at most $\lambda t\rho$. If we want the build of our barrier to succeed, we must make sure that all the vertices in our barrier are protected by a firefighter before the blob reaches the barrier. We will fail to do so if, at time $t = k$, the number of trees in R_k, N_k, is greater than the number of deployed firefighters fk. Therefore, we need the probability of failure to be bounded away from 1. To find an upper bound for this probability, we need to prove Lemma 5.1.

Lemma 5.1. *The probability that the number of trees in R_k, N_k, is greater than the number of deployed firefighters fk is bounded from above by $e^{-\frac{\delta^2\lambda\rho k}{3}}$, giving rise to* (5.1).

$$\mathbb{P}(N_k > fk) \le e^{-\frac{\delta^2\lambda\rho k}{3}} \tag{5.1}$$

Proof. We borrow from [36][Thm. 2.19, p. 70] the fact that, if $(X_i)_{i\ge1}$ is a sequence of independent identically distributed random variables, then, for all $a \ge \mathbb{E}[X_1]$, there exists a rate function $a \mapsto I(a)$ such that

$$\mathbb{P}\left(\sum_{i=1}^{n} X_i \ge na\right) \le e^{-nI(a)}. \tag{5.2}$$

Therefore, if we assume that (X_i) are independent Poisson random variables with mean $\mu_i = k\rho\lambda/n$ for every $i \ge 1$ and $a = (1+\delta)\mu_1$, we get

$$\mathbb{P}\left(\sum_{i=1}^{n} X_i \ge n(1+\delta)\mu_1\right) \le e^{-nI(a)}. \tag{5.3}$$

Moreover, since $f = 2\lambda(1+\delta)^2$ and $\rho = 2(1+\delta)$, we have that $fk = (1+\delta)k\rho\lambda$. Furthermore, $\sum_{i=1}^{n} X_i$ is the sum of independent Poisson random variables with mean $\mu_i = k\rho\lambda/n$ for every $i \ge 1$ and thus itself Poisson distributed with mean $\mu = k\rho\lambda$ [32], which is exactly the distribution of N_k, giving us (5.4).

$$\mathbb{P}\left(N_k \ge fk\right) \le e^{-nI(a)}. \tag{5.4}$$

We also borrow from [36][Thm. 2.19, p. 70] the fact that the rate function $a \mapsto I(a)$ can be computed as

$$I(a) = \sup_{t\ge0}\left(ta - \log\mathbb{E}\left[e^{tX_1}\right]\right). \tag{5.5}$$

To compute $I(a)$, we need $\mathbb{E}\left[e^{tX_1}\right]$, which we can compute as (5.6), using the definition for the expectation of a function of a random variable and for the probability of a Poisson random variable as in [21].

$$\mathbb{E}\left[e^{tX_1}\right] = \sum_{l=0}^{\infty} e^{tl} e^{-\mu_1}\frac{\mu_1^l}{l!} = e^{-\mu_1}\sum_{l=0}^{\infty}\frac{(\mu_1 e^t)^l}{l!} = e^{-\mu_1}e^{e^t\mu_1} = e^{\mu_1(e^t-1)} \tag{5.6}$$

To compute $I(a)$, we take a supremum over $t \ge 0$. This is equivalent to maximising $g(t) = ta - \log\mathbb{E}\left[e^{tX_1}\right]$ over $t \ge 0$, with $\mathbb{E}\left[e^{tX_1}\right]$ as in (5.6). To do so we need to find $t \ge 0$ such that $g'(t) = 0$, giving (5.7).

$$0 = g'(t) = \frac{\mathrm{d}}{\mathrm{d}t}\left(ta - \log\mathbb{E}\left[e^{tX_1}\right]\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(ta - \mu_1(e^t - 1)\right) = a - \mu_1 e^t. \tag{5.7}$$

From (5.7) we get $t = \log\left(a/\mu_1\right)$. There is indeed a maximum at this t, because $g''(t) = -\mu_1 e^t < 0$, assuming $\mu_1 > 0$. Therefore, we have

$$I(a) = a\log\left(\frac{a}{\mu_1}\right) - \mu_1(e^{\log(a/\mu_1)} - 1) = a\log\left(\frac{a}{\mu_1}\right) - a + \mu_1 I((1+\delta)\mu_1) \quad = \mu_1\left((1+\delta)\log(1+\delta) - \delta\right). \tag{5.8}$$

Note that to satisfy (5.1) we need

$$(1+\delta)\log(1+\delta) - \delta \geq \frac{\delta^2}{3} \tag{5.9}$$

We solve (5.9) numerically. If we define a function $f(\delta) = (1+\delta)\log(1+\delta) - \delta - \delta^2/3$ and use `fsolve(f, x0=1.7)[0]` in Python, we find the root of $f(\delta)$ to be roughly 1.81696. Therefore, for $\delta < 1.81696$, (5.9) and thus (5.1) is satisfied. To finish our proof, we note that $\mathbb{P}(N_k > fk) \leq \mathbb{P}(N_k \geq fk)$, implying (5.1). □

Proof. (Theorem 5.1) If E_k is the event that $N_k > fk$, then $\bigcup_{k=1}^{\infty} E_k$ is the event that $N_k > fk$ for every $k \geq 1$ and thus the event that we do **not** succeed in finishing the barrier and containing the fire. We have

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} E_k\right) \leq \sum_{k=1}^{\infty} E_k = \sum_{k=1}^{\infty} \mathbb{P}(N_k > fk) \leq \sum_{k=1}^{\infty} e^{-\frac{\delta^2\lambda\rho k}{3}} = \frac{e^{-\frac{\delta^2\lambda\rho}{3}}}{1 - e^{-\frac{\delta^2\lambda\rho}{3}}} \tag{5.10}$$

where we recognise the last sum as a geometric series [27]. For us to succeed in building our barrier and containing the fire, we need to bound the probability that we succeed away from zero and thus the probability that we do not succeed away from 1, meaning we need (5.10) to be smaller than 1. Suppose the distribution of trees in R is such that we do not succeed, then we repeat our calculations with an initial blob with radius $r_{\text{new}} > r_1$ whose corresponding teardrop is disjoint from the previous teardrop R. The event that this teardrop fails has probability bounded away from 1. We can repeat this process over and over again, each with probability of failure $p < 1$. The probability of ultimate failure after n tries is $\lim_{n\to\infty}\left(p\right)^n = 0$. Therefore, the method always works. It only rests us to bound (5.10). If we name $e^{-\delta^2\lambda\rho/3} = r$, we need $r/(1-r) < 1$, implying $r < 1/2$. This implies that we can contain the fire if $\lambda \geq \log\left(2^{2/3}\right)/(\delta^2 + \delta^3)$, where we use that $\rho = 2(1+\delta)$. This concludes the proof.

□

6

Lower bound for the blob

Bounding f_λ from below is not an easy task. Suppose that we wanted to prove $f_\lambda > c_1$ for some real number c_1, then we would have to check this bound for every possible firefighting strategy. For example, it is plausible that there are strategies that protect some key vertices in the midst of the fire and then surround it. The same difficulties are encountered if we look at the continuous case of surrounding a blob with a fence as in Chapter 4. However, if we assume that there are no fences inside the contained blob, we can find a lower bound. This gives rise to Theorem 6.1. We follow the exposition in [3].

Theorem 6.1. *For $\rho > 2$ the blob can be successfully surrounded by and end up as exactly the interior of a Jordan curve Φ of fence.*

We use Definition 6.1 to prove Theorem 6.1.

Definition 6.1. The *intrinsic diameter* is the supremum over all shortest paths between two distinct points $P \in \mathbb{R}^2$ and $Q \in \mathbb{R}^2$ contained in the interior of the curve Φ in $\mathbb{R}^2$.

Proof. (Theorem 6.1) Let R^* be the region of the contained blob and let D be the intrinsic diameter as defined in Definition 6.1. For every point $P \in \mathbb{R}^2$ in the interior of Φ its distance to the centre of the blob cannot exceed D, by Definition 6.1. Since the blob has unit growth rate and starts as a unit disk, the point P has been reached in time at most $D-1$. Therefore, the blob was surrounded at time less than D. Moreover, if we take two points $T \in \mathbb{R}^2$ and $Q \in \mathbb{R}^2$ on the fence Φ at distance D apart, the two parts of fence extending from $T \in \mathbb{R}^2$ to $Q \in \mathbb{R}^2$ on either side of the blob are of length at least D, since there would be a shorter shortest path otherwise. We can thus conclude that the building rate of the fence ρ, equal to the length of the fence divided by the building time, is more than 2: $\rho > 2$. □

Remark 6.1. Note that the lower bound obtained by Theorem 6.1 is exactly the upper bound derived in Chapter 4, meaning that the teardrop strategy is optimal when we only build around the blob and not inside it.

However, if we consider all possible fence-building strategies, we would hope to bound the building rate from below by 1. This is very hard to prove, but we can prove a similar theorem for an oriented blob, defined by Definition 6.2.

Definition 6.2. An *oriented blob* is a blob (Definition 4.2) that is restricted to grow only in a specified direction. More formally, the blobs growth rate function is only defined for $\theta \in [\theta_{\min}, \theta_{\max}]$ and at time $t = 0$ the blob is an arbitrary connected region $B(0) \subseteq \left\{[r\cos\theta, r\sin\theta]^\intercal : r \in [0,\infty), \theta \in [\theta_{\min}, \theta_{\max}]\right\}$, with $0 \le \theta_{\min} \le \theta_{\max} \le 2\pi$.

We can prove that for an oriented blob as in Definition 6.2, where $\theta_{\min} = \pi/2$ and $\theta_{\max} = \pi$, assuming the south is at angle $\theta = 0$, the critical fence-building rate ρ_c is at least 1, giving rise to Theorem 6.2.

Theorem 6.2. *The oriented blob (Definition 6.2) (and therefore the normal blob), where $\theta_{min} = \pi/2$ and $\theta_{max} = \pi$, assuming the south is at angle $\theta = 0$, cannot be contained by a fence built at rate $\rho \le 1$.*

Proof. We will prove by contradiction, following the exposition in [3]. Suppose that we have an oriented blob B^* that is contained by a fence built at rate $\rho_c \le 1$. Let $t = s$ be the time at which the blob is contained, giving $B^*(s)$ as the final blob. We let $(c,d) \in \partial B^*(s)$ be the point on the boundary of $B(s)$ furthest away from the

centre of the blob. We define the velocity vector $\vec{v}(t)$ of a point of the blob as in Definition 6.2. By our choice for $\theta_{\min}$ and $\theta_{\max}$, the x and y components of $\vec{v}(t,\theta,r)$ are non-negative, implying that the x and y coordinates of a point on the edge of the blob increase with time until the blob is contained. Therefore, the point (c,d) is the point where the sum of its x- and y-coordinates is maximised and we maximise $c+d$. We observe that c,d is the point in the most upper-right corner of the blob. Let $F(s)$ be the fence at time $t=s$ and let C represent all the horizontal parts of the fence and D all the vertical parts of the fence, such that $F(s) \supseteq C \cup D$. Note that we do not have to build fence west or south of the blob, since it cannot grow in this direction. If we project C onto the x-axis, it must cover the line segment $\{(x,0) : 0 \leq x \leq c\}$. This is in order for C to prevent the blob from growing in vertical direction. Therefore, the length of C must be at least c. Similarly, we derive that the length of D is at least d. Thus the length of the fence $F(s)$ is at least $c+d$. Moreover, since we assumed fence-building rate $\rho_c \leq 1$, it must have taken at least $c+d$ time to build the fence and hence s is at least $c+d$. Furthermore, we let P be the path taken by some point (a,b) in the blob to get to the point (c,d). Since we have an oriented blob, the path P is also oriented. This implies that the x and y coordinates of the points on the path P increase with time. Therefore, the longest path P is obtained if (a,b) is a point on the boundary and this path must have length at most $c+d-1$. Since the blob grows at unit growth rate, the path P must have been formed in time at most $c+d-1$, implying $s \leq c+d-1$. However, we also derived $s \geq c+d$, causing a contradiction. Hence, we have proved Theorem 6.2 by contradiction. □

7

Surrounding a blob with a fence influenced by wind

As described in [1] wind has a substantial impact on the propagation of fire. The fire propagates more rapidly in the direction of the wind, we call this region the head of the fire. The fire propagates slower in the direction opposite of the wind, we call this region the rear of the fire. We call the points with direction perpendicular to the wind the flanks of the fire. To get a more realistic fire propagation, we define a *blob with wind* in Section 7.1. We assume wind comes from the north. Since the rear of the blob propagates slower because of the wind and the rear of the blob without wind propagated with speed 1, we assume the growth rate function for the rear of the blob with wind can be at most 1. In contrast, since the front of the blob propagates faster because of the wind, we assume it propagates with growth rate function at least 1, subject to conditions specified in Definition 7.1. We assume the flanks of the blob propagate the same with or without wind. We propose a strategy for a blob with arbitrary wind in Section 7.3 and for a specific blob with wind in Section 7.2, where we assume an ellipse-shaped fire front.

7.1. Defining a blob with wind

We define a blob influenced by wind from the north.

Definition 7.1. A *blob with wind* is a blob (Definition 4.2) whose growth rate function $r(\theta)$ is defined by

$$r(\theta) = \begin{cases} r_1(\theta) & \text{for } \theta \in [0,\pi], \\ r_2(\theta) & \text{for } \theta \in [\pi, 3\pi/2], \\ r_2(3\pi-\theta) & \text{for } \theta \in [3\pi/2, 2\pi], \end{cases} \tag{7.1}$$

where $r_1 : [0,\pi] \to [0,1]$ is a continuous function with $r_1(0) = r_1(\pi) = 1$ and $r_2 : [\pi, 3\pi/2] \to [1, v_{\max}]$ is a continuous, increasing function with $r_2(\pi) = 1$ and $r_2(3\pi/2) = v_{\max}$, for some constant $v_{\max} \geq 1$. The initial blob is the set

$$\left\{ \begin{bmatrix} r(\theta)\cos(\theta) \\ r(\theta)\sin(\theta) \end{bmatrix} : \theta \in [0, 2\pi] \right\}.$$

7.2. Finding a teardrop-strategy for ellipse-shaped fire front

As an example of the blob with wind, we take an ellipse-shaped fire front. By this we mean that, for $\theta \in [\pi, 2\pi]$, $r(\theta)$ corresponds to the bottom half of the ellipse, defined by

$$x(t)^2 + \frac{y(t)^2}{v_{\max}^2} = (t+1)^2 \tag{7.2}$$

where $v_{\max} > 1$. We observe that the tip of the fire front propagates at speed $v_{\max}$ and the flanks still propagate at speed 1. We make no extra assumptions about the rear of the fire. The ellipse is a natural shape to consider after having considered a circular blob.

Theorem 7.1. *For the ellipse shaped fire-front we can enclose the blob with a teardrop-shaped fence built at rate $\rho > 2$.*

Proof. We will build two horizontal fences, both at building rate $\rho/2$, starting from $(0,-d)$ and going towards the west and east, where d is some constant to be determined. The blob will hit the fence at time $t = d/v_{\max}-1$ and since we need to build the fence below the initial blob we need $d \geq v_{\max}$. By similarity, we only show calculations for the fence built towards the east. Since we want the blob to grow along the fence, we substitute $y(t) = -d$ in (7.2), which gives

$$x_{\text{blob}}(t) = \sqrt{(t+1)^2 - \frac{d^2}{v_{\max}^2}} \tag{7.3}$$

for $t \geq d/v_{\max} - 1$. We observe that the function for $x_{\text{blob}}(t)$ does not depend on θ. We can now differentiate (7.3) with respect to t once to obtain the speed of the x-coordinate of the blob along the fence and twice to obtain its acceleration, giving us

$$\dot{x}_{\text{blob}}(t) = \frac{t+1}{\sqrt{(t+1)^2 - d^2/v_{\max}^2}} \tag{7.4}$$

and

$$\ddot{x}_{\text{blob}}(t) = \frac{-d^2/v_{\max}^2}{\left((t+1)^2 - d^2/v_{\max}^2\right)^{3/2}}. \tag{7.5}$$

Note that $\ddot{x}_{\text{blob}} < 0$. Therefore, $\dot{x}_{\text{blob}}(t)$ is a decreasing function. We want to bend our fence upwards, at time $t = s$ once the edge of the blob and the endpoint of the fence intersect. To do so we need to show that $x_{\text{blob}}(s) = s\rho/2$ and $\dot{x}_{\text{blob}}(s) \leq \rho/2$. Moreover, the blob should not pass the fence before time $t = s$, so we should check that $x_{\text{blob}}(t) \leq t\rho/2$ for all $t \in [d/v_{\max} - 1, s]$.

We observe that $\dot{x}_{\text{blob}}(t) = (t+1)/x_{\text{blob}}(t)$ and using $x_{\text{blob}}(s) = s\rho/2$ and $\dot{x}_{\text{blob}}(s) \leq \rho/2$, we obtain an inequality for s:

$$s \geq \frac{4}{\rho^2 - 4}. \tag{7.6}$$

To find d, we take minimum such s and substitute it into $x_{\text{blob}}(s) = s\rho/2$ to obtain

$$d = \frac{v_{\max}\rho}{\sqrt{\rho^2 - 4}}. \tag{7.7}$$

If we write $(x_{\text{blob}}(t), -d)$ in polar coordinates,let θ^* be the angle. We can take a look at the right triangle with opposite side d, adjacent side $s\rho/2$ and hypotenuse $\sqrt{d^2 + (s\rho/2)^2}$. This describes the situation at time $t = s$. We obtain

$$\theta^* = \tan^{-1}\left(\frac{d}{s\rho/2}\right) = \tan^{-1}\left(\frac{v_{\max}\sqrt{\rho^2-4}}{2}\right). \tag{7.8}$$

We observe that as ρ converges to 2, θ^* converges to $\tan^{-1}(0) = 0$. This implies that the radius of the blob at time s converges to 1 and therefore, just as with the regular blob, we can bend the fence into a teardrop-shape, enclosing the blob with an ellipse shaped fire front. It rests us to check that $x_{\text{blob}}(t) \leq t\rho/2$ is satisfied for all $t \in [d/v_{\max} - 1, s]$. By using (7.7) and rewriting $x_{\text{blob}}(t) \leq t\rho/2$, we obtain

$$\left(t - \frac{4}{\rho^2-4}\right)^2 \geq \left(\frac{4}{\rho^2-4}\right)^2 + \frac{4}{\rho^2-4} - \frac{d^2}{v_{\max}^2(\rho^2/4-1)} = 0 \tag{7.9}$$

which is true for all $t \in \mathbb{R}$. We can conclude that if we build a horizontal fence starting at $(0,-d)$, we can bend the fence upward at time $t = s = 4/(\rho^2-4)$ to enclose the blob with ellipse-shaped fire front. □

7.3. Finding a teardrop-strategy for arbitrary wind

We want to prove that the teardrop-strategy of Section 4.5 works for a blob with arbitrary wind. To make the strategy work, we need to build two horizontal fences towards the west and the east from $(0,-d)$. The fences must stay ahead of the edge of the blob and we bend the fence upwards into a teardrop shape when the blob's edge and the fence meet at time $t = s$. To enable us to bend the fence, we need the speed of the blob to not

exceed the fence building rate at the time of meeting. The goal is to find d and s so that the fence meets these conditions. To see the difficulty with an arbitrary blob with wind, we look back at the regular blob of Section 4.5. We built two horizontal fences at rate $\rho/2$ going west and east from $(0,-d)$. By similarity, we only looked at the part of the fence built towards the east, giving $x_{\text{fence}}(t) = t\rho/2$. The x-coordinate of the blob was described as $x_{\text{blob}}(t) = \sqrt{(t+1)^2 - d^2}$. We could then find s and d for which $x_{\text{blob}}(s) = x_{\text{fence}}(s)$ and $\dot{x}_{\text{blob}}(s) = \dot{x}_{\text{fence}}(s)$. To find the function $x_{\text{blob}}(t)$, we used the Pythagorean theorem. This was easy because $r(\theta)$ was constant for all $\theta \in [0, 2\pi]$, but if we have an (almost) arbitrary function for $r(\theta)$ such as with the blob with wind, $r(\theta)$ is not constant. Moreover, the radius of the blob at angle θ is $(t+1)r(\theta)$ and therefore x_{blob} depends on both time t and the angle θ, making it more difficult to find a formula for x_{blob}. Therefore, we need to find a new condition to check whether we can bend the fence.

Again, we only look at the horizontal part of the fence built towards the east. If we describe the intersection point of the blob with the fence in polar coordinates, let θ be the angle of this point. If the value of the growth rate function at angle θ, $r(\theta)$, would have been greater than $\rho/2$, the blob would have spilled over the fence and we could not have finished the teardrop-shaped fence. We leave the situation $r(\theta) = \rho/2$ as an open problem. Therefore, we need $r(\theta) < \rho/2$ for all $\theta \in \left(\theta_{\rho/2}, 2\pi\right]$, where $\theta_{\rho/2}$ is the angle for which $r(\theta) = \rho/2$. We have thus found a new condition for when to bend the fence. We can bend the horizontal fence, built at depth d, at time $t = s$, if $\theta \in \left[\theta_{\rho/2}, 2\pi\right]$.

7.3.1. Method for proving if the teardrop-strategy works for arbitrary blobs with wind.

We will now find the right angle θ^* for which $r(\theta^*) < \rho/2$ for an arbitrary blob with wind. We assume growth rate function $r(\theta)$ for all $\theta \in [3\pi/2, 2\pi]$. Furthermore, we assume that $r(\theta) < \rho/2$.

We can now use the Pythagorean theorem for time $t = s$, where the blob's edge and the fence meet, giving us

$$d^2 + s^2\rho^2/4 = (1+s)^2 r(\theta)^2 \tag{7.10}$$

which can be rewritten to a quadratic equation of s.

$$s^2\left(\rho^2/4 - r(\theta)^2\right) - 2sr(\theta)^2 + d^2 - r(\theta)^2 = 0 \tag{7.11}$$

We want (7.11) to have only one solution and thus set the discriminant D to 0. Solving $D = 0$ for d, we obtain d as a function of θ.

$$d(\theta) = \frac{r(\theta)\rho/2}{\sqrt{\rho^2/4 - r(\theta)^2}} \tag{7.12}$$

We can now solve (7.11) for s and plug in (7.12) to obtain s as a function of θ.

$$s(\theta) = \frac{r(\theta)^2}{\rho^2/4 - r(\theta)^2} \tag{7.13}$$

Note that if we had set $D > 0$ and solved (7.11) for s we would have obtained a negative (unvalid) s and a positive s larger than the s we found. Moreover, we would have obtained a smaller d than the d we found. We do not think a smaller d and at the same time larger s are very probable and therefore making the assumption $D = 0$ is a reasonable assumption. Looking at the right triangle with opposite side $d(\theta)$, adjacent side $s\rho/2$ and hypotenuse $(1+s)r(\theta)$, we can obtain an expression for the cosine of the angle that we call ϕ. By simplifying we obtain

$$\cos(\phi) = \frac{r(\theta)}{\rho/2} \tag{7.14}$$

To obtain the actual angle $\theta = \theta^*$ we need to find the intersection point of (7.15) with the function $\cos(\theta)$, which we can find by solving

$$\cos(\theta) = \frac{r(\theta)}{\rho/2} \tag{7.15}$$

for θ. However, there might not always be a solution $\theta^* \in \left[\theta_{\rho/2}, 2\pi\right]$ and this needs to be checked for the $r(\theta)$ that is used. Note that this is equivalent to checking $r(\theta^*) < \rho/2$ Assuming there is a solution, we use $d(\theta^*)$ as the depth of the horizontal fence and time $t = s(\theta^*)$ as the meeting time of the fence and the edge of the blob.

Still, the blob might have spilled over the fence before time $t = s$ and we need to check that

$$x_{\text{blob}}(t) \le t\rho/2 \quad \text{for all } t \in \left[\frac{d - v_{\max}}{v_{\max}}, s(\theta^*)\right] \tag{7.16}$$

where time $t = (d - v_{\max})/v_{\max}$ is the moment the blob hits the fence. Checking this condition for arbitrary $r(\theta)$ is a hard task, since it is difficult to find a formula for $x_{\text{blob}}(t)$. We leave the verification of (7.16) as an open problem.

Remark 7.1. We could have used the method for the arbitrary blob for the blob with the ellipse-shaped fire front. To do so, we need to use the explicit formula for $r(\theta)$ for an ellipse, which is

$$r(\theta) = \frac{v}{\sqrt{v_{\max}^2 \cos^2(\theta) + \sin^2(\theta)}} \tag{7.17}$$

However, we would need to get $r(\theta^*) = 1$ to get the same s and d for both methods, but solving $\cos(\theta) = r(\theta)/(\rho/2)$ for θ does not yield $\theta = \theta^*$ such that $r(\theta^*) = 1$. We therefore leave it as an open problem to prove that the method for arbitrary wind works.

8

Discussion/Conclusion

In this bachelor thesis, we looked at the firefighting problem on random geometric graphs. The firefighting problem is a problem on graphs where fire breaks out on a set of vertices. Each subsequent turn we can protect vertices using firefighters, after which the fire spreads to all unprotected neighbours of vertices on fire. The process ends when the fire cannot spread any more. The protected vertices and the vertices not on fire are the vertices saved. We explored the firefighting problem on random geometric graphs because these graphs much resemble actual rainforests and forest fires are increasing in frequency and intensity in recent years [40]. Vertices are placed randomly using a homogeneous Poisson Point Process. Vertices are connected by an edge if they are distance less than 1 apart. You could translate this to an actual forest where trees are not planted by humans but end up at the place where its seed, that was transported by the wind, landed. It makes sense that fire can only spread from trees next to each other and not between trees far apart. Although the firefighting problem on random geometric graphs is not an exact representation of forest fires, we hope that the development of this field of research can help fight forest fighters in the future.

We started in Chapter 2 by defining a random geometric graph. We gave some basic definitions about graphs before explaining the Homogeneous Poisson Point Process. Thereafter we defined the random geometric graph and some of its properties. In Chapter 3, we formally defined the firefighting problem and showed that we can find lower and upper bounds for the number of firefighters needed to contain the fire on a random geometric graph. We translated the firefighting problem on a random geometric graph to the problem of surrounding a continuously growing blob with a fence in Chapter 4. We defined the blob and the fence and explained the proof of [3] that we can surround the blob with a teardrop-shaped fence built at rate $\rho > 2$. We retranslated this result to the firefighting problem in Chapter 5, where we explained the proof of [3] that the number of firefighters needed to contain the fire on a random geometric graph is almost surely less than 2λ. In Chapter 6 we explained why it is difficult to bound the number of firefighters needed from below. We introduced the oriented blob and explained the proof of [3] that this blob and therefore the normal blob cannot be contained by a fence built at rate $\rho \leq 1$. In Chapter 7 we attempted to model firefighting in a more realistic way by introducing the *blob with wind*. Since, in the real world, the front of the fire propagates faster under the influence of wind and the rear of the fire propagates slower because of the wind, we introduced a blob having these exact properties. We proved that we can surround a blob with ellipse-shaped fire front, where the front of the blob propagates at speed $v_{\max}$, by a teardrop-shaped fence built at rate $\rho > 2$. We introduced a method that shows that the blob with arbitrary wind can (sometimes) also be surrounded by a teardrop-shaped fence built at rate $\rho > 2$.

We leave one major open problem resulting from this bachelor thesis. It would be interesting to formally prove the method that (if possible) determines how to surround an arbitrary blob with wind by a teardrop-shaped fence and to check that this method works for the blob with ellipse-shaped fire front.

There are multiple other interesting open problems concerning the firefighting problem. In [3] it is proved that for all $\lambda \geq 200$, if $f < \lambda/40$ then f is not sufficient to contain the fire on the the random geometric graph with density λ. It would be a great addition to the field to prove a general lower bound for the number of firefighters.

One might also consider different random graph models on which to study the firefighting problem. For instance, we might change the connection rules on the random geometric graph, so that the radii are chosen at random according to some distribution, or we might consider random graphs with weighted vertices. In

geometric inhomogeneous random graphs, vertices are assigned weights and vertices with high weights have higher degree. This could be interpreted as a vertex having more burning material and hence igniting more neighbouring trees.

Besides firefighting models where the underlying graph is random, there remain open problems where the underlying graph is a deterministic lattice. For example, the hexagonal or the triangular grid. Open problems in this setting are surveyed in [37] and we point out the following problem: does one firefighter suffice to contain a fire in the hexagonal grid? Partial progress has been made in [12].

Bibliography

[1] Angelo Alessandri, Patrizia Bagnerini, Mauro Gaggero, and Luca Mantelli. Parameter estimation of fire propagation models using level set methods. *Applied Mathematical Modelling*, 92:731–747, 2021.

[2] James Aspnes, Navin Rustagi, and Jared Saia. Worm versus alert: Who wins in a battle for control of a large-scale network? In *Principles of Distributed Systems; 11th International Conference, OPODIS 2007, Guadeloupe, French West Indies, December 17–20, 2007. Proceedings*, volume 4878 of *Lecture Notes in Computer Science*, pages 443–456. Springer-Verlag, 2007. doi: 10.1007/978-3-540-77096-1_32. URL https://doi.org/10.1007/978-3-540-77096-1_32.

[3] Amir Barghi and Peter Winkler. Firefighting on a random geometric graph. *Random Structures & Algorithms*, 46(3):466–477, 2013. ISSN 10429832. doi: 10.1002/rsa.20511. URL https://onlinelibrary.wiley.com/doi/10.1002/rsa.20511.

[4] Anurag Bishnoi. Graph theory.

[5] Béla Bollobás and Oliver Riordan. *Percolation*. Cambridge University Press, Cambridge, 2006.

[6] Anthony Bonato and Richard J. Nowakowski. *The Game of Cops and Robbers on Graphs*, volume 61 of *Student Mathematical Library*. American Mathematical Society, 2011. ISBN 978-0-8218-5347-4. doi: 10.1090/stml/061. URL https://bookstore.ams.org/view?ProductCode=STML%2F61. Available in softcover and eBook formats.

[7] Anthony Bonato, Margaret-Ellen Messinger, and Paweł Prałat. Fighting constrained fires in graphs. *Theoretical Computer Science*, 434:11–22, 2012.

[8] Alberto Bressan. Dynamic blocking problems for a model of fire propagation. *Advances in Applied Mathematics, Modeling, and Computational Science*, pages 11–40, 2013.

[9] Andrea Burgess, John Marcoux, and David Pike. Firefighting with a distance-based restriction. *arXiv preprint arXiv:2204.01908*, 2022.

[10] Leizhen Cai, Yongxi Cheng, Elad Verbin, and Yuan Zhou. Surviving rates of trees and outerplanar graphs for the firefighter problem. *preprint*, 2009.

[11] Leizhen Cai, Yongxi Cheng, Elad Verbin, and Yuan Zhou. Surviving rates of graphs with bounded treewidth for the firefighter problem. *SIAM Journal on Discrete Mathematics*, 24(4):1322–1335, 2010.

[12] Abdullah Dean, Sean English, Tongyun Huang, Robert A Krueger, Andy Lee, Mose Mizrahi, and Casey Wheaton-Werle. Firefighting on the hexagonal grid. *Discrete Applied Mathematics*, 305:16–22, 2021.

[13] Alexander Dean, Sean English, Tongyun Huang, Robert A Krueger, Andy Lee, Mose Mizrahi, and Casey Wheaton-Werle. Firefighting on the hexagonal grid and on infinite trees. *arXiv preprint arXiv:2010.05060*, 2020.

[14] Louis Esperet, Jan Van den Heuvel, Frédéric Maffray, and Félix Sipma. Fire containment in planar graphs. *Journal of Graph Theory*, 73(3):267–279, 2013.

[15] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Randomized broadcast in networks. *Random Structures & Algorithms*, 1(4):447–460, 1990. ISSN 1042-9832. doi: 10.1002/rsa.3240010406. URL https://onlinelibrary.wiley.com/doi/10.1002/rsa.3240010406.

[16] Stephen Finbow and Gary MacGillivray. The firefighter problem: a survey of results, directions and questions. *Australas. J Comb.*, 43:57–78, 2009.

[17] Stephen Finbow, Andrew King, Gary MacGillivray, and Romeo Rizzi. The firefighter problem for graphs of maximum degree three. *Discrete Mathematics*, 307(16):2094–2105, 2007.

[18] A. M. Frieze and G. R. Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10:57–77, 1985. ISSN 0166-218X. doi: 10.1016/0166-218X(85)90004-2. URL https://doi.org/10.1016/0166-218X(85)90004-2.

[19] A. Ganesh, L. Massoulié, and D. Towsley. The effect of network topology on the spread of epidemics. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1455–1466. IEEE, 2005. doi: 10.1109/INFCOM.2005.1498374. URL https://ieeexplore.ieee.org/document/1498374.

[20] Tomáš Gavenčiak, Jan Kratochvíl, and Paweł Prałat. Firefighting on square, hexagonal, and triangular grids. *Discrete Mathematics*, 337:142–155, 2014.

[21] Geoffrey Grimmett and Dominic Welsh. *Probability: An Introduction*. Oxford University Press, 2nd edition, 2014. ISBN 978-0198709978. ISBN-13: 978-0198709978, ISBN-10: 0198709978.

[22] B. L. Hartnell. Firefighter! an application of domination. Presentation at the Twenty-Fifth Manitoba Conference on Numerical Mathematics and Computing, University of Manitoba, 1995.

[23] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988. ISSN 0028-3045. doi: 10.1002/net.3230180406. URL https://onlinelibrary.wiley.com/doi/10.1002/net.3230180406.

[24] Andrew King and Gary MacGillivray. The firefighter problem for cubic graphs. *Discrete Mathematics*, 310(3):614–621, 2010.

[25] Jiangxu Kong, Weifan Wang, and Xuding Zhu. The surviving rate of planar graphs. *Theoretical Computer Science*, 416:65–70, 2012.

[26] Jiangxu Kong, Lianzhu Zhang, and Weifan Wang. The surviving rate of digraphs. *Discrete Mathematics*, 334:13–19, 2014.

[27] Stephen R. Lay. *Analysis with an Introduction to Proof*. Pearson, Amsterdam, 2020. ISBN 9781784342821. Compiled by Technische Wiskunde, Delft University of Technology for TW1010 and TW1040.

[28] Cai Leizhen and Wang Weifan. The surviving rate of a graph for the firefighter problem. *SIAM Journal on Discrete Mathematics*, 23(4):1814–1826, 2010.

[29] Gary MacGillivray and Shayla Redlin. The firefighter problem on orientations of the cubic grid. *Bull. ICA*, 88:22–29, 2020.

[30] M Messinger. Firefighting on the triangular grid. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 63:37, 2007.

[31] Paweł Prałat. Graphs with average degree smaller than burn slowly. *Graphs and Combinatorics*, 30(2): 455–470, 2014.

[32] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, 10th edition, 2010. ISBN 9780123756862. URL https://www.amazon.com/Introduction-Probability-Models-Sheldon-Ross/dp/0123756863.

[33] Sudip Saha, Abhijin Adiga, and Anil Vullikanti. Equilibria in epidemic containment games. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 777–783. AAAI Press, 2014. doi: 10.1609/aaai.v28i1.8819. URL https://doi.org/10.1609/aaai.v28i1.8819.

[34] Stack Overflow Community. Stack overflow badges, n.d. URL https://i.sstatic.net/lu2JS.png. Accessed: 2025-06-10.

[35] James Stewart. *Calculus: Early Transcendentals*. Cengage Learning, Boston, MA, 8th ed edition, 2016. ISBN 978-1-305-30653-7.

[36] Remco van der Hofstad. *Random Graphs and Complex Networks: Vol. I*. Cambridge University Press, Cambridge, 2016. ISBN 9781107172859. URL https://www.win.tue.nl/~rhofstad/NotesRGCN.html.

[37] Connor Wagner. *A new survey on the firefighter problem.* PhD thesis, 2021.

[38] Weifan Wang, Xubin Yue, and Xuding Zhu. The surviving rate of an outerplanar graph for the firefighter problem. *Theoretical Computer Science*, 412(8-10):913–921, 2011.

[39] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. In *Proceedings of the 22nd International Symposium on Reliable Distributed Systems (SRDS 2003)*, pages 25–34. IEEE, 2003. doi: 10.1109/RELDIS.2003.1238052. URL `https://ieeexplore.ieee.org/document/1238052`.

[40] World Resources Institute. Global trends in forest fires, 2024. URL `https://www.wri.org/insights/global-trends-forest-fires`. Accessed: 2025-06-15.