Radboud University Nijmegen

Faculty of Science

The Critical Value of the Contact Process

Bachelor Thesis in Mathematics

Author:
Casper Algera

Supervisor:
dr. Henk Don

Second reader:
dr. TBD TBD

May 19, 2025

Contents

1 Prerequisites

Throughout this thesis, we will be making connections to a variety of stochastic processes. To keep the text around my own results compact, I have collected all definitions and notation for these processes in this first section. To start, we need to establish what a stochastic process is. Any process that evolves randomly over time, is considered a stochastic process.

Definition 1.1. A stochastic process is a collection of random variables $\{X_t\}_{t\in A}$. The process is called discrete-time if A is a finite or countable set and continuous-time if A is uncountable.

In the remainder of this chapter we will introduce various stochastic processes that we will use to study the contact process, as well as some broader results from probability theory.

1.2 The Galton-Watson branching process

One of the stochastic processes we will consider in this thesis is the Galton-Watson branching process.

Definition 1.3. The Galton-Watson process is a stochastic process $\{X_n\}_{n\in\mathbb{N}]}$ with the X_n i.i.d. such that $X_n : \mathbb{N} \mapsto \mathbb{R}$. We maintain a list of nodes, starting with v_0. For each node in the list of nodes, the random variable X_n determines the number of children of the nth node, these children get added onto the list of nodes.

The nodes as generated by the Galton-Watson process described above generate a tree by drawing edges between the children of each node. This tree is called a Galton-Watson tree.

Example 1.4. An example of a Galton-Watson process with the discrete uniform offspring distribution $X_n \sim \text{Unif}\{1, 2, 3\}$ is illustrated in Figure 1. We start with v_0, and generate children according to X_0. This add two children, v_1 and v_2. We go down the list to the next node, v_1, which generates children according to X_1. This continues either indefinitely, or until all nodes in the list have generated children.

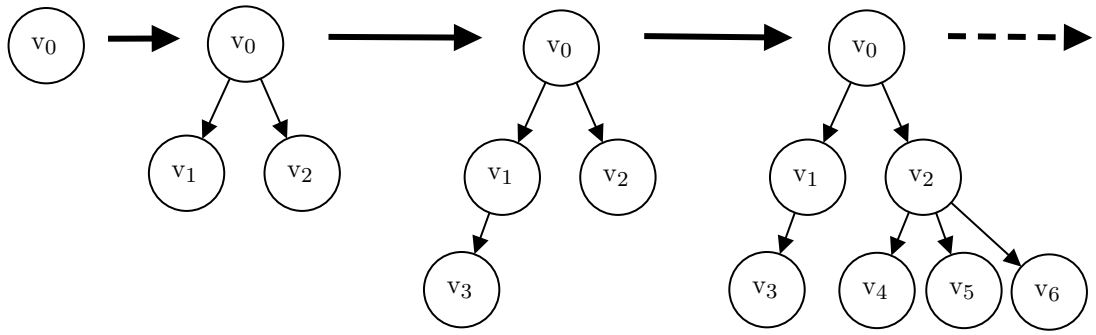


Figure 1: Example of a Galton-Watson process

Note that in this example, the process is bound to continue indefinitely, since we always generate at least one node. Similarly, if we take X_n to be zero with probability 1, the process will always be finite. It turns out that there is a very general result about the survival of this process.

Let Z_n denote the number of individuals in the nth generation, we want to know whether or not the process goes extinct, that is, whether or not there exists an N such that $Z_N = 0$. This is neatly summarised in the following theorem.

Theorem 1.5. *For a branching process with offspring distribution* X*, the process dies out if and only if* $\mathbb{E}[X] \leq 1$ *and* $\mathbb{P}[X = 1] < 1$*.*

The full proof of this theorem is rather involved, primarily because of the case that $\mathbb{E}[X] = 1$. This case is not imperative for the results of this thesis, so we will provide an proof which is much simpler, that only show that the process dies out asymptotically almost surely if $\mathbb{E}[X] < 1$ and has a positive probability of surviving if $\mathbb{E}[X] > 1$.

Proof. This proof primarily relies on the fact that

$$\begin{aligned}\mathbb{E}[Z_n] &= \sum_{m\in\mathbb{N}} \mathbb{E}[Z_n \mid Z_{n-1} = m]\,\mathbb{P}[Z_{n-1} = m] \\ &= \sum_{m\in\mathbb{N}} m\,\mathbb{E}[X]\,\mathbb{P}[Z_{n-1} = m] \\ &= \mathbb{E}[X]\,\mathbb{E}[Z_{n-1}].\end{aligned}$$

So that, by induction,

$$\mathbb{E}[Z_n] = \mathbb{E}[X]^n.$$

Now suppose that $\mathbb{E}[X] < 1$, then $\lim_{n\to\infty} \mathbb{E}[Z_n] = \lim_{n\to\infty} \mathbb{E}[X]^n = 0$. Thus, $\lim_{n\to\infty} \mathbb{P}[Z_n \geq 1] \leq 0$ by the Markov inequality. Hence, $\mathbb{P}[Z_n = 0] = 1$.

If $\mathbb{E}[X] > 1$, we will show that there exists a fixed $\varepsilon > 0$ such that there exists a N in $\mathbb{N}$ such that for all $k \geq N$ we have that $\mathbb{P}[Z_n \geq k] = \mathbb{P}[X^n \geq k] > \epsilon$. Since $\mathbb{E}[X] > 1$ there exists an ε such that $\mathbb{P}[X > 1] > 2\varepsilon$. Now since

$$\lim_{n\to\infty} \mathbb{P}[X^n \geq N] = \lim_{n\to\infty} \mathbb{P}[X \geq \sqrt[n]{N}] = \mathbb{P}[X > 1] > 2\varepsilon,$$

it must hold that there exists a N such that for all $k \geq N$ we have that $\mathbb{P}[Z_k \geq n] > \epsilon$. This shows that the tree dies out with probability strictly smaller than 1, proving the desired result. □

The complete proof of this theorem, can be found in [8, p. 86], alongside a many more results on branching processes. The study of these processes has many more interesting questions, but these are of not of relevance to this thesis. If you want to read more, you can refer to [8].

1.6 Percolation processes

Percolation models are stochastic processes the were initially devised to model the flow of a liquid through a porous medium. We will look at bond percolation, where one assigns a set of nodes and edges, and give each possible edge a probability p to be open.

Definition 1.7. The percolation process on a graph G is a collection of Bernoulli random variables with parameter p given by $(X_e)_{e\in E_G}$, where E_G denotes the edges of G.

The interpretation of Definition 1.7 is that we get a new graph with the same nodes as G and all the edges where $X_e = 1$. After we have done this process we ask if there exists an certain path through which water can flow. In infinite graphs, we look for an infinitely long path. We call this event "percolation". If we do this percolation process on an infinite graph G, we can define a so called critical probability.

Definition 1.8. The critical probability $p_c(G)$ for percolation on an infinite graph G is given by

$$p_c(G) = \inf\{\ p \mid \mathbb{P}[\{\text{percolation}\}] > 0\}.$$

We also often assign an origin, denoted by $\underline{0}$. Then the event that there is an infinite path passing through the origin is denoted by $\mathbb{P}_p\left[\{\underline{0} \leftrightsquigarrow \infty\}\right]$. In particular, it is important to note that $\{\underline{0} \leftrightsquigarrow \infty\} \subset \{\text{percolation}\}$.

One of the processes that we will be interested in oriented bond percolation on $\mathbb{Z}^2_{\geq 0}$. This is the same percolation process as in Definition 1.7, but with positively oriented edges. That is, horizontal edges go from (i, j) tot $(i + 1, j)$ and vertical edges go from (i, j) tot $(i, j + 1)$.

The a part of the oriented percolation process on $\mathbb{Z}^2$ is illustrated in Figure 2. A path to the edge of the picture has been marked in green. Percolation implies that there is a strictly positive probability that there exists a green path like this which is infinitely long.

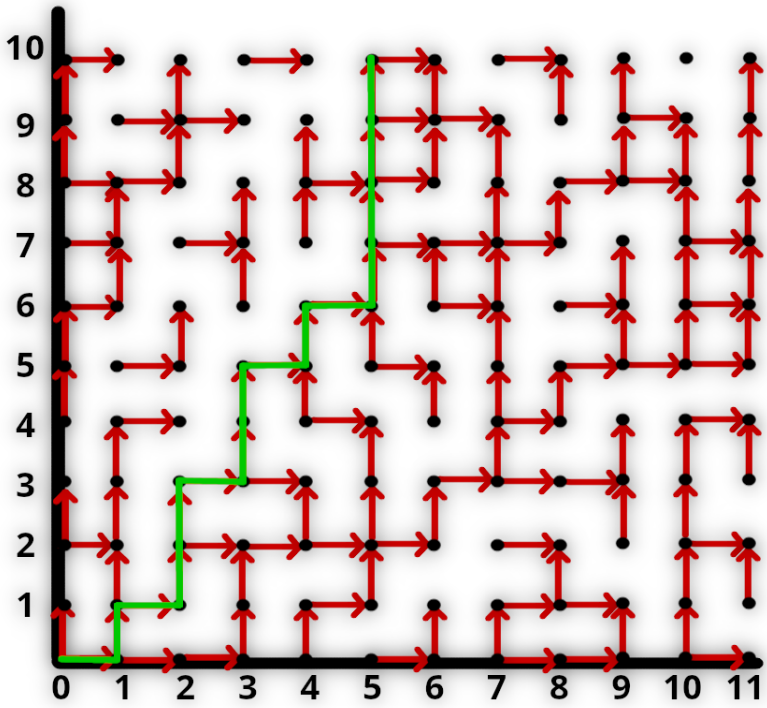


Figure 2: Infection path in oriented percolation

The study of percolation is one that is rich of interesting problems. However, for the purposes of this thesis, this short introduction suffices. If one wants to learn more about percolation, a good source is Geoffrey Grimmett's book [3].

1.9 Results from probability theory

Throughout this thesis, we will often consider exponential distributions. We will also sporadically use some other results from probability theory that might require a refresher. All these results have been collected in this subsection. First we recall what a memoryless distribution is.

Definition 1.10. We say that a random variable X is memoryless if

$$\mathbb{P}[X > t + a \mid X > a] = \mathbb{P}[X].$$

Example 1.11. Suppose that $X \sim \text{Exp}(\lambda)$, then X is memoryless, since

$$\mathbb{P}[X > t + a \mid X > a] = \int_a^{t+a} \frac{\lambda e^{-\lambda x}}{e^{-\lambda a}}\, dx = \int_a^{t+a} \lambda e^{-\lambda(x-a)} dx = \int_0^t \lambda e^{-\lambda x} dx = \mathbb{P}[X].$$

One useful result is that every (continuous-time) memoryless distribution is exponentially distributed.

Theorem 1.12. *If a continuous-time random variable* X *is memoryless then there exists a* $\lambda \in \mathbb{R}$ *such that* $X \sim \text{Exp}(\lambda)$.

As outlined in [4, p. 68], this is a direct consequence of the fact that the only real-valued function that satisfies $f(s+t) = f(s)f(t)$ is the exponential function. By the nature of the contact process, it is often useful to compare two exponentially distributed random variables.

Lemma 1.13. *If* $X_1 \sim \text{Exp}(\lambda_1)$ *and* $X_2 \sim \text{Exp}(\lambda_2)$ *then* $\mathbb{P}[X_1 \geq X_2] = \dfrac{\lambda_1}{\lambda_1 + \lambda_2}$.

One way to prove this entails integrating the joint distribution, which is the product of the individual distributions by independence, over all values where $X_1 \geq X_2$. For the complete proof the reader can refer to [6, p. 292]. Another elementary tool from probability theory that will prove useful is the union bound.

Lemma 1.14. *For events* $A_1, \ldots, A_n$,

$$\mathbb{P}\left[\bigcup_{i=1}^{n} A_i\right] \leq \sum_{i=1}^{n} \mathbb{P}[A_i].$$

2 Introduction to the contact process

In this section we introduce the subject of study in this thesis, the contact process. We start by discussing Markov processes in general. Next, we define the contact process and cover some important definitions and initial results. Finally, we discuss the self-duality property.

2.1 Markov processes

When modelling problems with stochastic processes we often deal with situations where the time you have to wait until the next event is memoryless. That is, regardless of how long you have been waiting, the probability of the event occurring within some fixed timespan remains constant. We can think of examples like the time until the next earthquake or until the failure of a machine.

This motivates the introduction of a special class of processes called Markov processes. In loose terms, a process is Markovian if the next state of the process does not depend on its history.

Definition 2.2. We say that a continuous-time stochastic process $\{X_t\}_{t\geq 0}$ with a countable state space S has the Markov property if

$$\mathbb{P}[X_t = x \mid X_r,\, 0 \leq r \leq s] = \mathbb{P}[X_t = x \mid X_s]$$

for $x \in S$ and $t \geq s \geq 0$.

In this thesis we will primarily consider Markov processes on graphs. Each node will have a state, and transitions to another state based on local rules. Another property we will come across is called time-homogeneity. In essence, time-homogeneity means that the transition probabilities do not depend on time.

Definition 2.3. We say that a continuous-time stochastic process $\{X_t\}_{t\geq 0}$ with state space S is time-homogenous if

$$\mathbb{P}[X_t = x \mid X_s = y] = \mathbb{P}[X_{t-s} = x \mid X_0 = y]$$

for $x, y \in S$.

One example of a Markov process is a continuous time Markov chain. This is a process with a discrete state space, where we describe the transitions from one state to the next. To ensure the Markov property, the transitions must be exponentially distributed due to Theorem 1.12. Hence, it suffices to give the parameter for each of the exponential distributions, which we also call the transition rate.

Definition 2.4. A continuous-time Markov chain is a continuous-time Markov process $(X_t)_{t\geq 0}$ with $X_t \in V$. It has a countable state space S with transitions rates $(s_{i,j})_{i,j\in S}$ corresponding to the rate at which X_t is state v_i transitions to v_j.

The prime example of a Markov process is the Poisson process. This process is discussed in great detail in many texts on undergraduate probability, such as [4, p. 65] and [6, p. 292]. For completeness, one description of the Poisson process is included below.

Example 2.5. We look at a series of events where the times between occurrences are distributed by $T_i \sim \text{Exp}(\lambda)$. Let $N_t = \max\left[n : \sum_{i=0}^{n} T_i \leq t\right]$, this is what we call the Poisson process. We claim that this is a Markov process, which is also time-homogenous. First recall from Example

1.12 that the transition times are memoryless. Using this, it is evident that this is a time-homogenous Markov process. The only thing that impacts the future probabilities is how many events there have been so far. It does not matter what time the previous event occurred at. The Poisson process can be thought of as a continuous-time Markov chain with state space $\mathbb{N}$ with transition rates $n \to n+1$ equal to λ.

2.6 The contact process

We start with the definition of the contact process.

Definition 2.7. The contact process is a continuous-time Markov Process on the space of subsets of nodes of an undirected graph $G = (V, E)$ with a parameter λ, where the transitions are determined by

$$\begin{aligned} &\text{for every } x \in \xi_t, \quad \xi_t \mapsto \xi_t \setminus \{x\} \text{ with rate } 1, \\ &\text{for every } x \notin \xi_t, \quad \xi_t \mapsto \xi_t \cup \{x\} \text{ with rate } \lambda|\{y \in \xi_t \colon \{x, y\} \in E\}|. \end{aligned} \tag{2.1}$$

In this model we consider a node to be healthy if it is in state 1 and infected if it is in state 0. To fully define this process, we need to specify an initial occupancy. That is, we need to specify which nodes are infected at $t = 0$. We will write $\{\xi_t\}_{t\geq 0}$ if we we start with full occupancy. To specify a contact process with initial occupancy $A \subset V$ we use $\{\xi_t^A\}_{t\geq 0}$.

Throughout this thesis we will be studying the contact process on the square lattice on $\mathbb{Z}^d$. This is a graph with nodes in $\mathbb{Z}^d$, and edges between any pair of points that has Manhattan distance exactly 1. We refer to this graph simply by $\mathbb{Z}^d$, unless it can reasonably be confused with the set $\mathbb{Z}^d$. We will primarily concern ourselves with the so-called critical infection rate $\lambda_c(\mathbb{Z}^d)$. This is the value of λ at which there occurs a fundamental change in the limiting behaviour of the stochastic process, analogous to critical probabilities in the study of random graphs.

Definition 2.8. The critical infection rate on a infinite connected graph G, is defined by

$$\lambda_c(G) = \inf\left\{\lambda : \lim_{t\to\infty} \xi_t^A = \mathbf{0}\right\}. \tag{2.2}$$

For any finite subset A of the vertices of G.

Heuristically, we can think of the critical infection rate as the smallest rate that ensures that the infection dies out. It might not be entirely clear why this would even be a well-defined quantity, this is a question we will come to later, proving it in the case $G = \mathbb{Z}^d$. The quantity is also well-defined for other graphs, but the proof is more involved and not relevant to this thesis. Let us now we consider finite graphs. In this case the infection always dies out, so the critical value is trivially 0. Hence, it makes sense to look at another metric, namely the extinction time, which is defined as follows.

Definition 2.9. The extinction time of the contact process of a graph G with initial occupancy $A \subset G$ is given by

$$\tau_G^A = \inf\{t : \xi_t^A = \mathbf{0}\}. \tag{2.3}$$

When talking about the contact process we distinguish the so called subcritical and supercritical regions. These are the value of λ such $\lambda < \lambda_c(G)$ and $\lambda > \lambda_c(G)$ respectively. Qualitatively, we expect the share of infected sites to quickly tend towards 0 in the subcritical case, whereas in the supercritical case we expect the share of infected nodes to hover certain number nodes. This illustrated below in Figure 3, which graphs the amount of infections over time in a simulation of

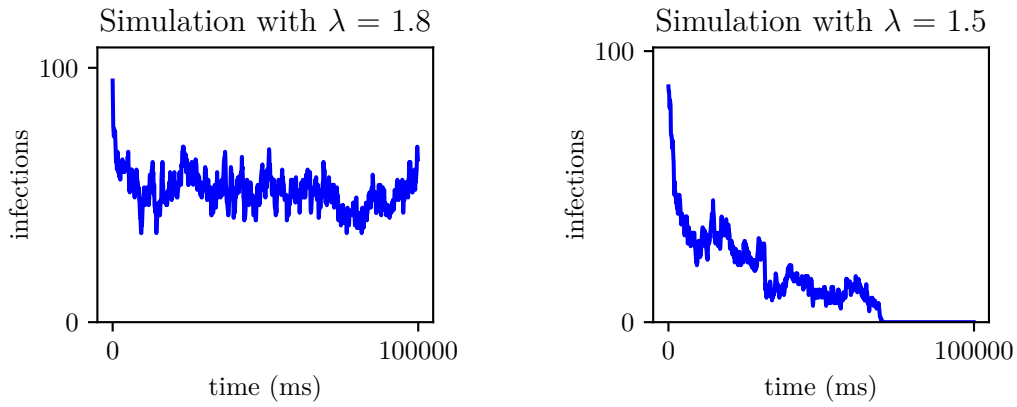


Figure 3: The contact process in the sub- and supercritical case

the contact process on a cyclic graph of length 100. We would expect similar behaviour on the graph $\mathbb{Z}$ where all nodes are infected at the start.

There exist a few different characterisations of the critical value of the contact process. We first define the survival function.

Definition 2.10. The survival function $\psi : \mathbb{R}_{\geq 0} \mapsto [0, 1]$ maps a parameter λ of the contact process $\xi_t^{\{0\}}$ on $\mathbb{Z}^d$ to $\mathbb{P}\left[\lim_{t\to\infty} \xi_t^{\{0\}} = \mathbf{0}\right]$.

This survival function is often an object of study in literature regarding the contact process, and will prove to be essential later in this chapter. In the final chapter we will develop methods for estimating the critical value, which also allow us to visualise this function.

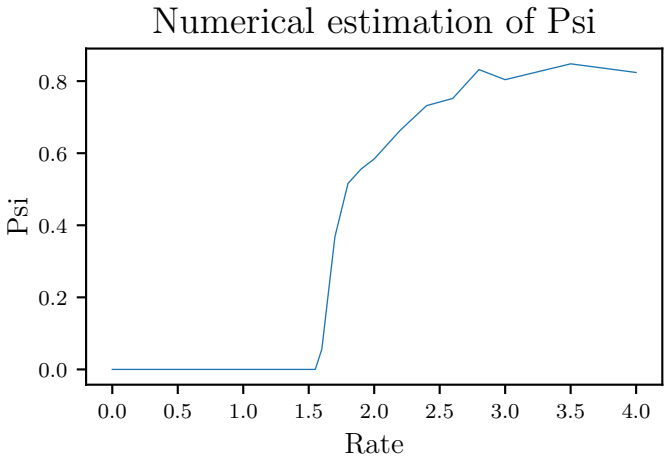


Figure 4: Numerical estimation of psi

2.11 Couplings

In this thesis we will greatly rely on the following two definitions. In order to study the contact process, it is often useful to make a comparison to a simpler process. We will often do so using a technique called coupling.

Definition 2.12. A coupling between two random variables $X_1 : \Omega_1 \mapsto \mathbb{R}$ and $X_2 : \Omega_2 \mapsto \mathbb{R}$ is a new probability space Ω' with two random variables $X_1' : \Omega' \mapsto \mathbb{R}$ and $X_2' : \Omega' \mapsto \mathbb{R}$ such that X_1' and X_2' are identically distributed to X_1 and X_2 respectively.

Most comparison we will want to make is where one process dominates the other, that is, if an event happen in one process, it also happens in the other. This motivates the following definition.

Definition 2.13. A random variable $X : \Omega \mapsto \mathbb{R}$ is said to (stochastically) dominate $Y : \Omega \mapsto \mathbb{R}$ if

$$\mathbb{P}[Y \geq x] \leq \mathbb{P}[X \geq x], \text{ for all } x \in \mathbb{R}.$$

If this is the case, we write $X \leq_{st} Y$. If we have that

$$\mathbb{P}[Y \geq x] < \mathbb{P}[X \geq x], \text{ for all } x \in \mathbb{R}.$$

We say that X strictly (stochastically) dominates Y, and we write $X <_{st} Y$

This in itself is not yet sufficient, as we will often we dealing with random variables that do not necessarily lie in the same probability space. The following result, shows that it is sufficient to find a suitable coupling.

Lemma 2.14. *A random variable $X : \Omega_1 \mapsto \mathbb{R}$ is said to stochastically dominate $Y : \Omega_2 \mapsto \mathbb{R}$ if and only if there exists a coupling (X', Y') on Ω' such that $\mathbb{P}[X' \geq Y'] = 1$.*

For the proof the reader can refer to [8, p. 64]. Note that these definitions rely on a simple notion of probability spaces. One could easily formulate these in terms of measure theory, but we will refrain from doing so where possible.

It would be worth studying this proof

One very important result on the contact process is due to Harris. There exists a coupling between the contact process and a carefully selected set of Poisson processes. For each node in the graph we assign a rate 1 Poisson process, and a rate λ Poisson process for each outgoing edge. The rate 1 process determines the time when a site becomes healthy. The rate λ processes determine when the process infects the corresponding neighbour, if they're infected themselves. For the contact process on $\mathbb{Z}$, an illustration can be found in Figure 5. In Figure 5 the blue crosses represent recoveries, and red arrows corresponds to infections. In the figure we can see that the node –1 is infected at time T if node 0 is infected at time 0, since we draw a path from 0 at time 0 to –1 at time T without any recoveries. We will mimic notation from percolation, writing $v_1 \leftrightsquigarrow \infty$ if there is an infinitely long infection path in time starting in v_1 at time 0.

Let us now return to the question of whether or not Definition 2.8 is well defined for $G = \mathbb{Z}^d$. The answer is positive, by the following lemma.

Lemma 2.15. *Considering a contact process $(\xi_t)_{t\geq 0}$ on a graph G. Then the following are equivalent*

1. $\psi(\lambda) > 0$
2. $\lambda > \lambda_c$

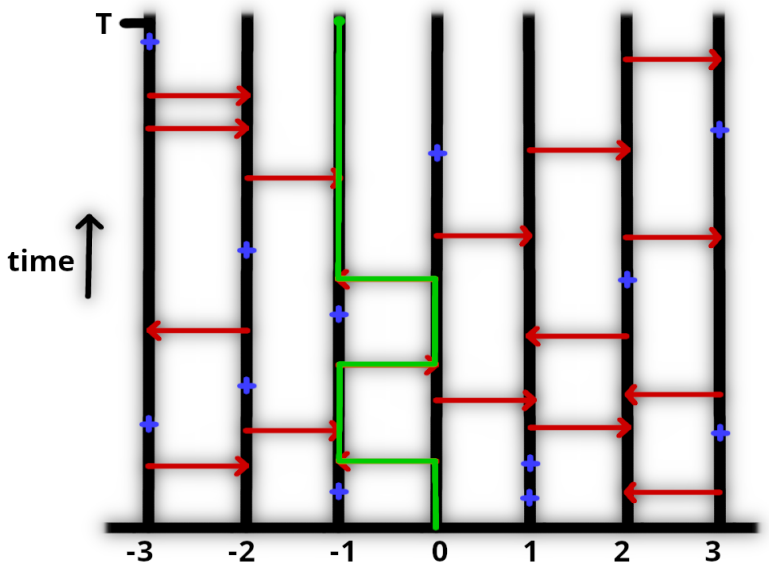


Figure 5: Harris' visual representation

Proof. The implication (1) $\implies$ (2) is trivial. Now let us assume by contraposition that $\psi(\lambda) = 0$. Let Ξ be one realisation of Harris' visual representation. Then

$$\begin{aligned}\mathbb{P}\left[\lim_{t\to\infty} \xi_t^A = \mathbf{0}\right] &= \mathbb{P}\left[A \leftrightsquigarrow \infty \text{ in } \Xi\right] \\ &\leq \sum_{i\in A} \mathbb{P}\left[i \leftrightsquigarrow \infty \text{ in } \Xi\right] \\ &= \sum_{i\in A} \mathbb{P}\left[\lim_{t\to\infty} \xi_t^{\{i\}} = \mathbf{0}\right] = 0\end{aligned}$$

using the union bound (Lemma 1.14). This shows that $\lambda \leq \lambda_c$, completing the proof. □

In addition to showing that Definition 2.8 is well defined for $G = \mathbb{Z}^d$, Lemma 2.15 allows use to study the much simpler contact process with one initial infection to conclude things about the critical probability. We will make use of this fact extensively, without explicitly mentioning this lemma.

2.16 Self duality

3 The contact process on finite graphs

In this first section we will briefly look at the contact process on n-cyclic graphs. In particular, we will derive a way to find a crude upper and lower bound on the expected extinction time. Additionally, we will take a look at general finite graphs, and prove strict inequalities on extinction time, for graphs that are strictly included in a another graph.

3.1 Expected extinction time on cyclic graphs

In this section we will derive a crude upper and lower bound on the expected extinction time of the contact process on n-cyclic graphs. We will do so by considering the probabilities that a certain number of nodes are infected at a set time. These probabilities are very hard to determine, but we are able to find an upper and lower bound. Since we are dealing with continuous-time Markov chains, we need to derive some further results on the exponential distribution.

Lemma 3.2. *If* X_1 *and* X_2 *are independently exponentially distributed with parameters* λ_1 *and* λ_2 *respectively, then* $\min(\mathrm{X}_1, \mathrm{X}_2) \sim \mathrm{Exp}(\lambda_1 + \lambda_2)$.

Proof. Since X_1 and X_2 are independent,

$$\mathbb{P}[\min(\mathrm{X}_1, \mathrm{X}_2) > \mathrm{a}] = \mathbb{P}[\mathrm{X}_1 > \mathrm{a}, \mathrm{X}_2 > \mathrm{a}] = \mathbb{P}[\mathrm{X}_1 > \mathrm{a}]\,\mathbb{P}[\mathrm{X}_2 > \mathrm{a}] = \mathrm{e}^{-\lambda_1 \mathrm{a} - \lambda_2 \mathrm{a}}.$$

Hence $\mathbb{P}[\min(\mathrm{X}_1, \mathrm{X}_2) <= \mathrm{a}] = 1\text{–}\mathrm{e}^{-\lambda_1 \mathrm{a} - \lambda_2 \mathrm{a}}$, and since this is precisely the cumulative distribution function of $\mathrm{Exp}(\mathrm{X}_1 + \mathrm{X}_2)$, it must hold that $\min(\mathrm{X}_1, \mathrm{X}_2) \sim \mathrm{Exp}(\lambda_1 + \lambda_2)$. □

This now allows us to prove the following lemma, which will be useful for deriving the main result of this section.

Lemma 3.3. *Given that* $\mathrm{X}_1 \sim \mathrm{Exp}(\lambda_1)$ *and* $\mathrm{X}_2 \sim \mathrm{Exp}(\lambda_2)$ *independently*

$$\mathbb{E}[\mathrm{X}_1 \mid \mathrm{X}_1 < \mathrm{X}_2] = \tfrac{1}{\lambda_1 + \lambda_2}.$$

Proof. We start by noting that the X_1 is equal to $\min(\mathrm{X}_1, \mathrm{X}_2)$ under the condition that $\mathrm{X}_1 < \mathrm{X}_2$, hence

$$\begin{aligned} \mathbb{E}[\mathrm{X}_1 \mid \mathrm{X}_1 < \mathrm{X}_2] &= \mathbb{E}[\min(\mathrm{X}_1, \mathrm{X}_2)] \\ &= \frac{1}{\lambda_1 + \lambda_2} \end{aligned}$$

Using that $\min(\mathrm{X}_1, \mathrm{X}_2) \sim \mathrm{Exp}(\lambda_1 + \lambda_2)$ as proven in Lemma 3.2. □

Now that we have this tool we can formulate and prove the main results of this section. We will determine the lower and upper bound by comparing with a suitable continuous time Markov chain. Let us first determine a lower bound.

Theorem 3.4. *Let* $\{\xi_{\mathrm{t}}^{\mathrm{A}}\}_{\mathrm{t} \geq 0}$ *denote the contact process be a contact process on the graph* $\mathrm{G} = \mathrm{C}_{\mathrm{n}}$*, that is, an* n*-cyclic graph. Then a lower bound* X_{i} *of* $\mathbb{E}[\tau_{\mathrm{G}}^{\mathrm{A}} \mid \mathrm{A}_{\mathrm{i}}]$ *can be found for all* $1 \leq \mathrm{i} \leq \mathrm{n}$ *by solving the system*

$$\begin{cases} \mathrm{X}_1 &= \frac{1}{(2\lambda+1)^2} + \frac{2\lambda}{2\lambda+1}\left(\mathrm{X}_2 + \frac{1}{2\lambda+1}\right), \\ \mathrm{X}_{\mathrm{i}} &= \frac{\mathrm{i}}{2\lambda+\mathrm{i}}\left(\mathrm{X}_{\mathrm{i}-1} + \frac{1}{2\lambda+\mathrm{i}}\right) + \frac{2\lambda}{2\lambda+\mathrm{i}}\left(\mathrm{X}_{\mathrm{i}+1} + \frac{1}{2\lambda+\mathrm{i}}\right),\ 2 \leq \mathrm{i} \leq \mathrm{n}-1, \\ \mathrm{X}_{\mathrm{n}} &= \mathrm{X}_{\mathrm{n}-1} + \frac{1}{\mathrm{n}}. \end{cases}$$

Where A_{i} *denotes the event that* $|\mathrm{A}| = \mathrm{i}$.

Proof. We want to establish a lower bound on the number of infections. We establish an order on the edges and on the nodes of C_n. Observe the the total rate of new infections across the whole graph C_n, is at least 2λ if there is an healthy site. We will couple a continuous Markov chain in Figure 6 to the contact process on C_n so that if the contact process has gone extinct, the chain has as well. Any time step from i to $i-1$ in the Markov chain corresponds to a recovery in one of the first i infected nodes in the contact process. A step from i to $i+1$ in the Markov chain

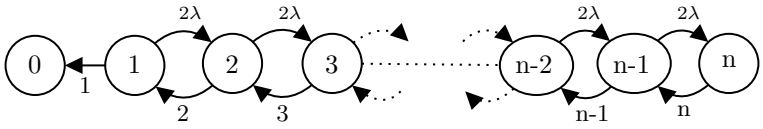


Figure 6: Lower bound Markov chain

corresponds to a infection through one of the first two edges between a healthy and an infected site. Any other events in the contact process do not affect the Markov chain. By construction, the state of this Markov chain is certainly smaller or equal to the number of infections in the contact process, so the time until this Markov reaches 0 starting at $|A|$ is stochastically dominated by τ_G^A.

Now, we can derive a system of equations based on this Markov chain. Let $X \sim \exp(2\lambda)$ and $Y_i \sim \exp(i)$, then given that we are in state $2 \le i \le n-1$ we can find the expected time until extinction based on the neighbouring states. We let X_i be the time that this Markov chain goes extinct if we start in state i, then the expected value of X_i satisfies

$$\begin{cases} X_1 &= \mathbb{P}[Y_1 < X]\,\mathbb{E}[Y_1 \mid Y_1 < X] + \mathbb{P}[X < Y_1]\left(X_2 + \mathbb{E}[X \mid X < Y_1]\right), \\ X_i &= \mathbb{P}[Y_i < X]\left(X_{i-1} + \mathbb{E}[Y_i \mid Y_i < X]\right) + \\ & \quad \mathbb{P}[X < Y_i]\left(X_{i+1} + \mathbb{E}[X \mid X < Y_i]\right),\ 2 \le i \le n-1 \\ X_n &= X_{n-1} + \mathbb{E}[Y_n]. \end{cases}$$

By Lemma 1.13 we get $\mathbb{P}[Y_i < X] = \dfrac{i}{2\lambda + i}$ and $\mathbb{P}[X < Y_i] = \dfrac{2\lambda}{2\lambda + i}$ and by Lemma 3.3 we find $\mathbb{E}[X \mid X < Y_i] = \mathbb{E}[Y_i \mid Y_i < X] = \dfrac{1}{2\lambda + i}$ to get the desired result. □

Now, to find an upper bound we will analogously determine a Markov chain that gives an upper bound on the number of infections.

Theorem 3.5. *Let $\{\xi_t^A\}_{t\ge 0}$ denote the contact process be a contact process on the graph $G = C_n$, that is, an n-cyclic graph. Then a lower bound X_i of $\mathbb{E}[\tau_G^A \mid A_i]$ can be found for all $1 \le i \le n$ by solving the system*

$$\begin{cases} X_1 &= \frac{1}{(2\lambda+1)^2} + \frac{2\lambda}{2\lambda+1}\left(X_2 + \frac{1}{2\lambda+1}\right), \\ X_i &= \frac{i}{2i\lambda+i}\left(X_{i-1} + \frac{1}{2i\lambda+i}\right) + \frac{2i\lambda}{2i\lambda+i}\left(X_{i+1} + \frac{1}{2i\lambda+i}\right),\ 2 \le i \le \left\lfloor \frac{n-1}{2} \right\rfloor, \\ X_i &= \frac{i}{2(n-i-1)\lambda+i}\left(X_{i-1} + \frac{1}{2(n-i-1)\lambda+i}\right) + \\ & \quad \frac{2(n-i-1)\lambda}{2(n-i-1)\lambda+i}\left(X_{i+1} + \frac{1}{2(n-i-1)\lambda+i}\right),\ \left\lfloor \frac{n-1}{2} \right\rfloor < i \le n-1, \\ X_n &= X_{n-1} + \frac{1}{n}. \end{cases}$$

Where A_i denotes the event that $|A| = i$.

Proof. We no consider a Markov chain which follows the upper bounds on the infection rates. The infection rate is maximal when there are as many healthy sites bordering infectious ones as possible. While the number of infections is strictly smaller than $\frac{n}{2}$, this means we have an infection rate of at most 2λ times the number of infected sites. Once the number of infections is larger or equal to $\frac{n}{2}$ we get a rate of at most 2λ times the number of healthy sites. We will use this fact to couple to the continuous-time Markov chain in Figure 7. We can couple the Markov

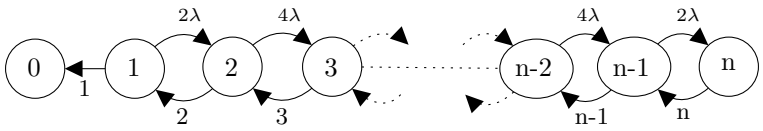


Figure 7: Upper bound Markov chain

chain starting at $|A|$ in Figure 7 to a contact process starting with initial infections A. Let $r_{1,t}$ be the total infection rate in our contact process at time t with initial configuration A, and $r_{2,t}$ be the rate from i tot $i+1$ in our Markov chain at time t. Let s denote the time of the last event, and S denote the state of the Markov chain. Wait until there is an infection or recovery in the contact process, until $T_1 \sim \text{Exp}(r_{2,s} - r_{1,t})$ seconds have passed or until $T_2 \sim \text{Exp}(S - |\xi_s|))$, then

- Let there be a step from i to $i+1$ if there was an infection in the contact process
- Let there be a step from i to $i-1$ if there was an recovery in the contact process
- Let there be a step from i to $i+1$ if we waited until T_1
- Let there be a step from i to $i-1$ if we waited until T_2

By construction, the state of this Markov chain is certainly greater or equal to the number of infections in the contact process, so the time until this Markov reaches 0 starting at $|A|$ stochastically dominates τ_G^A. The rest of this proof of this theorem is analogous to that of Theorem 3.4, by again deriving a system of equations from the Markov chain.

□

For small values of n, you can write computer programs that are capable of solving these systems algebraically. We have done so to visualise the quality of these bounds, the scripts utilised can be found in Appendix A.1. If we specifically look at the extinction time given full occupancy, we get something that looks like Figure 8. We can see that as n increases, the difference between our upper and lower bound increases drastically. This pattern holds across all initial configurations, but gets slightly less drastic.

With the aid of the numeric solver an attempt has been made to derive and explicit solution for the systems in this chapter, but this was to no avail. As introduced in Figure 3, there is a change in behaviour in infinite graphs around the critical value. Heuristically, one might think a similar change would occur in finite graphs. To investigate this, we will look at solutions of the system for fixed values of λ as n grows.

Typically the bounds like the ones rate 1, illustrated in Figure 9a. The upper bound seemingly grows exponentially, whilst the lower bound looks to grow logarithmically. Consequently, it is unclear how the expected extinction time behaves as n grows. However if we now look at the graph with rate 0.25 in Figure 9b up until $n = 40$, we will see that both the upper and lower bound appear to grow logarithmically. Additionally, the bound is actually quite tight for these smaller values of λ.

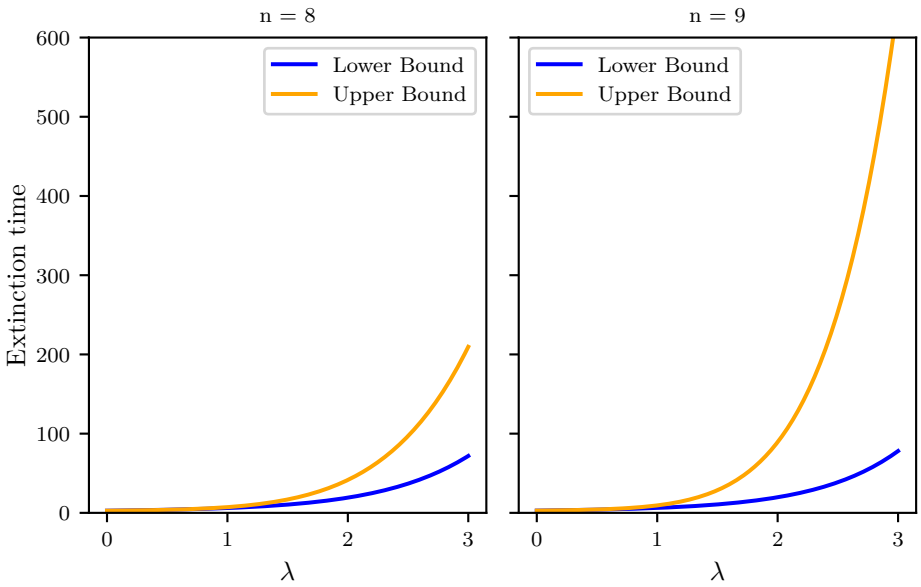


Figure 8: Upper and lower bound for n = 8 and n = 9

Additionally, if we take the rate greater than $\frac{n}{2}$, we see exponential growth until n in the lower bound. This is illustrated in Figure 9c, for rate 9 up until n = 19, to staying within the floating point range. This hints at the fact that a change in behaviour also occurs in these finite graphs, however since the change does not occur in the lower bound for a fixed value of λ we cannot definitively say that it does in fact occur.

3.6 Strict inequalities on extinction time

One might also wonder if we can prove any strict (stochastic) inequalities on the extinction time of more general classes of graphs. The first result in the section concerns finite graphs that are strictly included in another. To get there, we will first show the (non-strict) inequality.

Theorem 3.7. *If* $G_1 \subset G_2$ *are graphs:*

- *If* G_1 *and* G_2 *are finite, then* $\tau^A_{G_1} \leq_{st} \tau^A_{G_2}$ *for all non-empty initial occupancies* A
- *If* G_1 *and* G_2 *are infinite, then* $\lambda_c(G_1) \leq \lambda_c(G_2)$.

Proof. We proof this statement by defining a suitable coupling. Define a contact process of G_2 according to Harris' visual representation. That is, we had a rate 1 Poisson process that determines recoveries for every node, and a rate λ Poisson process for every edge, that determines infections. Since $G_1 \subset G_2$, we can define a contact process on G_1, using these Poisson processes, by letting each node and edge use the respective rate 1 and rate λ Poisson processes.

If the infection survives in G_1, we then immediately know that it must also survive in G_2, allowing us to apply Lemma 2.14 to conclude that the indicator of the event of survival in G_2 dominates that of G_1. Consequently, we get that $\tau^A_{G_1} \leq_{st} \tau^A_{G_2}$ for all non-empty initial occupancies A in the finite case and hence that $\lambda_c(G_2) \leq \lambda_c(G_1)$ in the infinite case. □

Given this inequality, it suffices to show that the random variables $\tau^A_{G_1}$ and $\tau^A_{G_2}$ are different, for all A and $G_1 \subsetneq G_2$. To do so, we will identify a strictly positive probability, that G_1 goes extinct, while G_2 survives for some time after the extinction time of G_1.

Theorem 3.8. *If $G_1 \subsetneq G_2$ are finite connected graphs, then $\tau^A_{G_1} <_{st} \tau^A_{G_2}$ for all non-empty initial occupancies.*

Proof. Since we have a finite graph, the total rate of new events is always smaller or equal to $r = 2\lambda|E_{G_2}| + |V_{G_2}|$ where $|E_{G_2}|$ and $|V_{G_2}|$ denote the number of edges and vertices of G_2 respectively. Throughout this proof, we utilise the same coupling as used in Theorem 3.7. Additionally, because of this same theorem, it suffices to prove $\tau_{G_1} \neq \tau_{G_2}$.

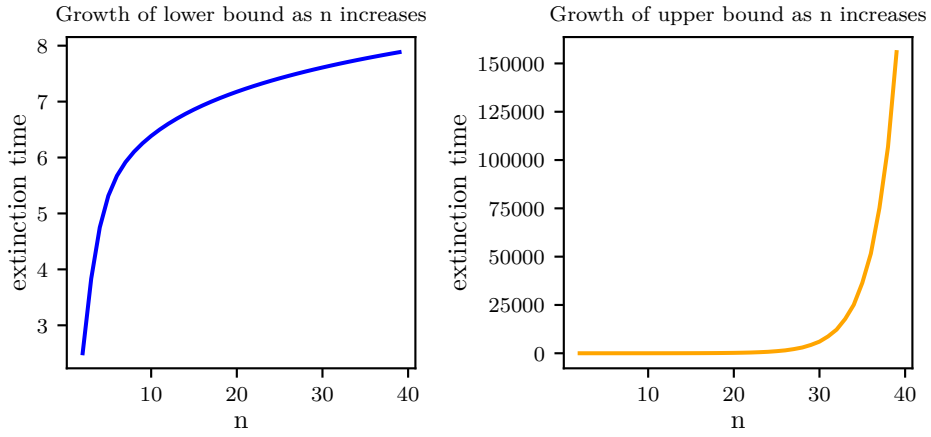
We first show that, without loss of generality, we can assume that the initial occupancy is full occupancy. We do this by showing that there is a positive probability to reach full occupancy, after which $\tau_{G_1} \neq \tau_{G_2}$ implies $\tau^A_{G_1} \neq \tau^A_{G_2}$ by the Markov property of the contact process. Connectedness of the graph implies that there is always a probability of infection greater or equal to $\frac{\lambda}{r}$ if there are uninfected sites. Let E denote the event that $(\xi^A_t)_{t\geq 0}$ reaches full occupancy starting from $t = 0$, then

$$\mathbb{P}[E] \geq \prod_{i=1}^{|V_{G_2}\setminus A|} \frac{\lambda}{r} > 0$$

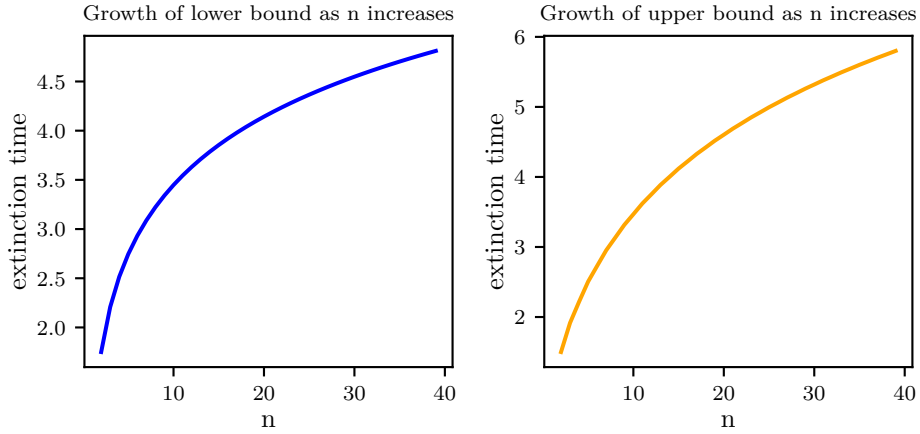
Now assuming full occupancy, we will use the same trick to show that there is a strictly positive probability that all sites of G_1 recover at at time smaller then T, whilst none of $G_2 \setminus G_1$ do, all the while there are no infections. The probability that a node recovers before any new infections is therefore greater or equal to $\frac{|\xi_t|}{r}$ by Lemma 1.13. This gives

$$\mathbb{P}[\tau_{G_1} < \tau_{G_2}] \geq \prod_{i=1}^{|V_{G_1}|} \frac{i}{r} > 0,$$

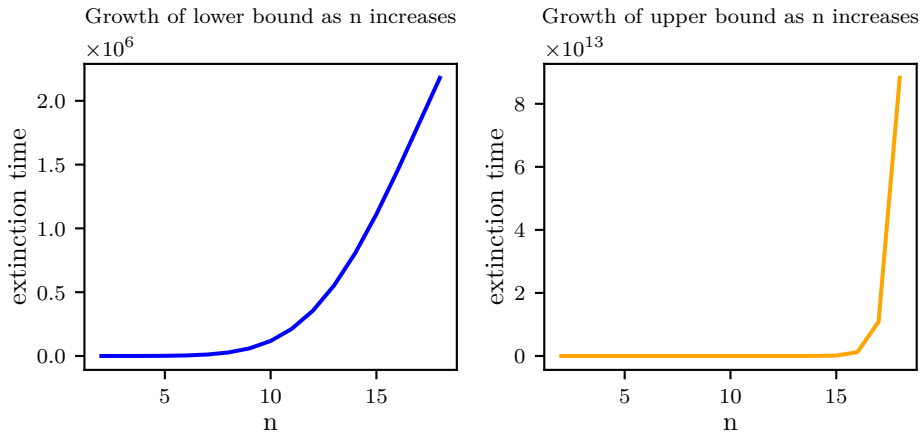
proving that $\tau_{G_1} \neq \tau_{G_2}$, completing the proof. □



(a) Growth of upper and lower bound for rate 1



(b) Growth of upper and lower bound for rate 0.25



(c) Growth of upper and lower bound for rate 9

Figure 9: Growth of upper and lower bounds for various rates

4 Upper and lower bounds on the critical value

In this section we will derive upper and lower bounds for $\lambda_c(\mathbb{Z})$.

4.1 Lower bound

We can obtain a first asymptotic lower bound by making a coupling to Galton-Watson trees. We will will explore the contact process until a certain number of the changes have occurred. To do so we define "frontier events".

Definition 4.2. We say there is a frontier event at time t in the contact process anytime there is a recovery of an infected node, or recovery of an healthy node.

In order to couple with Galton-Watson trees we need to ensure the independence of the child distributions. We can cleverly exploit the memoryless property of the Poisson processes to check all infections paths of the contact process we are interested in without creating any dependence, by switching to a new process anytime we would be overlapping an explored path.

Lemma 4.3. *The contact process survives if the expected number of children after* k *frontier events is less or equal to* 1 *and* $\mathbb{P}$[*exactly 1 child after* k *events*] < 1

Proof. We will couple a contact process $\{\xi_t\}_{t\geq 0}^{\{0\}}$ with a suitable Galton-Watson tree. We take our probability space to be $\left(\left(\{\xi_t\}_{t\geq 0}^{\{0\}}\right)_n\right)_{n\in\mathbb{N}}$, which are $\mathbb{N}$ distinct ordered copies of the contact process. We will do a suitable exploration of these processes, to embed a Galton-Watson tree in this space. We define the set of parents I_n which stores pairs (x, t) which are infected nodes x at time t in generation n. We start with one infection, and thus $I_0 = \{(x, 0)\}$. Start by marking all times for every node in every process as unexplored. Then, for each infected node (x, t_0) in I_n, we find the children as follows.

1. Set the number of exploration steps i equal to 0.
2. Let B be the set of children for this parent. We start with $B = \{0\}$
3. Now, while B is non-empty and $i < k$ we repeat the following.
   (a) For each $y \in B$ we will consider the smallest contact process $\left(\{\xi_t\}_{t\geq 0}\right)_{j_y}$ that has been not been explored at time t_0. Now we will continue exploring until the next frontier event, or the smallest t such that there exists an $\left(\{\xi_t\}_{t\geq 0}\right)_{j_y}$ that has been explored at node y in time t.
      - In the first case we denote the current time by t_1, increment i by 1 and mark every y in B as explored in $\left(\{\xi_t\}_{t\geq 0}\right)_{j_y}$ for times $[t_i, t_{i+1}]$. Finally, we update B according the the recovery or infection that took place.
      - In the second case we mark every y in B as explored in $\left(\{\xi_t\}_{t\geq 0}\right)_{j_y}$ from t to the current time and continue from (a) with t_0 equal to the current time.
4. Finally, for each $y \in B$ add (y, t) to I_{n+1} where t is the time at which the exploration ends.

We can repeat this for each generation, which is illustrated in Figure 10. Note that switching between the exploring different contact processes, causes no problems, since both the recoveries and infections are memoryless. The construction precisely gives a Galton-Watson tree, where

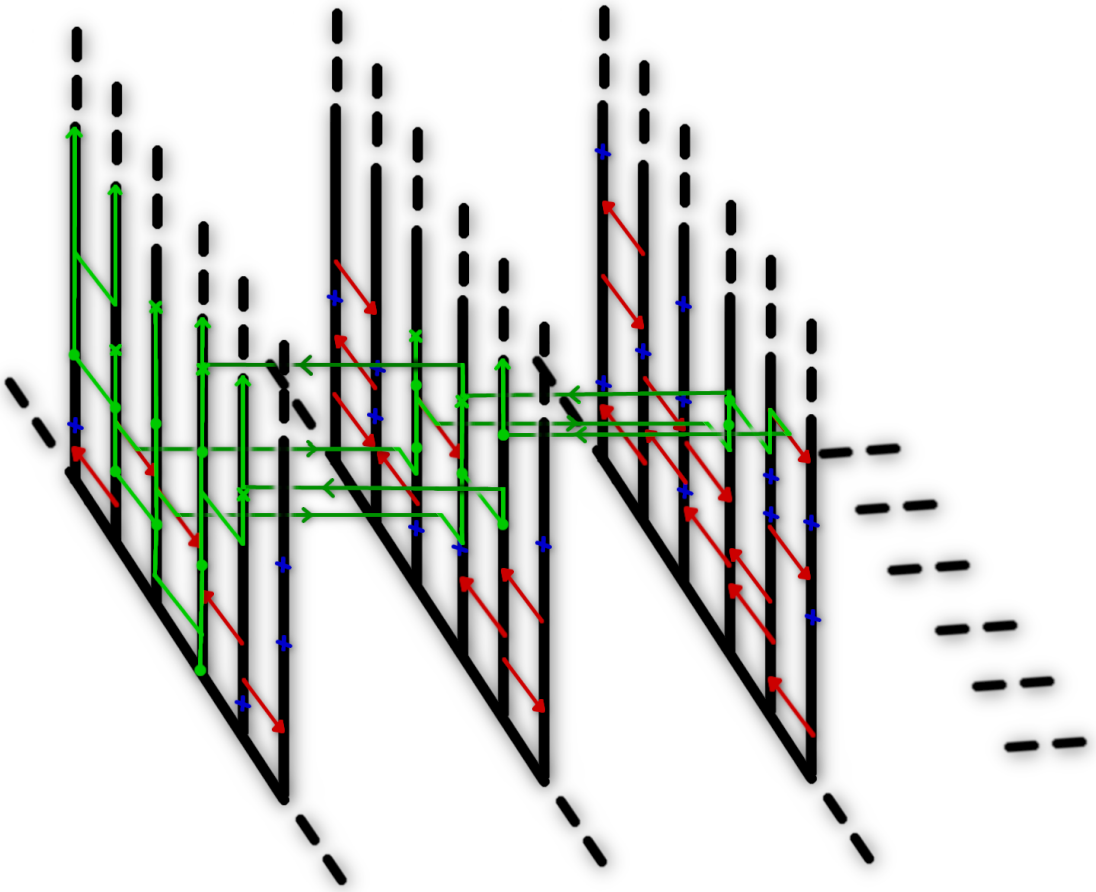


Figure 10: Coupling with Galton Watson trees

the children are determined by the number of children after k frontier events. Now it remains to show that if the Galton-Watson tree dies out, then the contact process dies out. Assume, by contraposition, that the contact process survives. Then there exists an infinite path in the contact process, starting in x. Since we are always exploring the primary contact process first, that must mean that the Galton-Watson tree survives, completing the proof. □

For small values of k, it is possible to exactly determine the distribution of children. This allows us to establish an asymptotic lower bound.

Theorem 4.4. *For all* $\mathrm{d} \in \mathbb{N}$ *we have that* $\lambda_c\left(\mathbb{Z}^{\mathrm{d}}\right) \geq \frac{1}{8} + \frac{\sqrt{18\mathrm{d}-1}}{8\mathrm{d}\sqrt{2\mathrm{d}-1}}$.

Proof. We will determine the expected number of children after 2 frontier events, starting with 1 infection. Denote this by X. The probability that the infection remains after 1 frontier event is $\frac{2\mathrm{d}\lambda}{2\mathrm{d}\lambda+1}$, in which case there will be two infected sites. Either these sites infect another site, with probability $\frac{2(2\mathrm{d}-1)\lambda}{2(2\mathrm{d}-1)\lambda+2}$ or they heal with probability $\frac{2}{2(2\mathrm{d}-1)\lambda+2}$. Here we use that a cluster of two infected sites has $2(2\mathrm{d}-1)$ uninfected neighbours. Now we calculate the expected number

children.

$$\begin{aligned}\mathbb{E}[X] &= \frac{2d\lambda}{2d\lambda+1}\left(\frac{2}{2(2d-1)\lambda+2}+3\frac{2(2d-1)\lambda}{2(2d-1)\lambda+2}\right)\\ &= \frac{2d\lambda}{2d\lambda+1}\frac{6(2d-1)\lambda+2}{2(2d-1)\lambda+2}\\ &= \frac{12d(2d-1)\lambda^2+4d\lambda}{4d(2d-1)\lambda^2+2(2d-1)\lambda+4d\lambda+2}\end{aligned}$$

Now we need to solve $\mathbb{E}[X] \leq 1$, writing out gives

$$\begin{aligned}&12d(2d-1)\lambda^2+4d\lambda \leq 4d(2d-1)\lambda^2+2(2d-1)\lambda+4d\lambda+2\\ &4d(d-1)\lambda^2-(2d-1)\lambda-1 \leq 0.\end{aligned}$$

Now, by solving this quadratic we find that the infection survives if

$$\begin{aligned}\lambda &\geq \frac{2d-1+\sqrt{(2d-1)^2+16d(2d-1)}}{8d(2d-1)}\\ &= \frac{2d-1+\sqrt{36d^2-20d+1}}{8d(2d-1)}\\ &= \frac{1}{8}+\frac{\sqrt{(2d-1)(18d-1)}}{8d(2d-1)}\\ &= \frac{1}{8}+\frac{\sqrt{18d-1}}{8d\sqrt{2d-1}}.\end{aligned}$$

Which finally allows us to conclude using Lemma 4.3 that $\lambda_c\left(\mathbb{Z}^d\right) \geq \frac{1}{8}+\frac{\sqrt{18d-1}}{8d\sqrt{2d-1}}$. □

4.5 Upper bound

From Theorem 3.7 we can immediately conclude the following statement.

Corollary 4.6. *For all* $d \in \mathbb{N}$ *we have that* $\lambda_c(\mathbb{Z}^{d+1}) \leq \lambda_c(\mathbb{Z}^d)$

This implies that an upper bound of the contact process on $\mathbb{Z}$ gives an upper bound for all $\mathbb{Z}^d$. We can do much better however. The following is a result by Richard Holley and Thomas Liggett. We follow the proof given in [5, p.307].

Theorem 4.7. *(R. Holley, T. M. Liggett, 1978) Let* $\{\xi_t\}_{t\geq 0}$ *by the contact process on* $\mathbb{Z}^d$ *with parameter* λ *and* $\{\zeta_t\}_{t\geq 0}$ *the contact process on* $\mathbb{Z}$ *with parameter* λd*. Then,*

1. $$\mathbb{P}[\xi_t^{\{0\}} \neq \varnothing] \leq \mathbb{P}[\zeta_t^{\{0\}} \neq \varnothing]$$

   for all $t \geq 0$.

2. $d\lambda_c\left(\mathbb{Z}^d\right) \leq \lambda_c\left(\mathbb{Z}\right)$

Proof. To proof 1, we define a function $\pi_d \colon \mathbb{Z}^d \mapsto \mathbb{Z}$ by

a implies b using self-duality

$$\pi_d(x_1,\ldots,x_d) = x_1+\cdots+x_d,$$

and for any finite set A,

$$\pi_{\mathrm{d}}(\mathrm{A}) = \{\pi_{\mathrm{d}}(\mathrm{x}) \mid \mathrm{x} \in \mathrm{A}\}.$$

We will now make a coupling between $\{\xi_{\mathrm{t}}\}_{\mathrm{t}\geq 0}$ and $\{\zeta_{\mathrm{t}}\}_{\mathrm{t}\geq 0}$ that ensures that

$$\zeta_{\mathrm{t}} \subset \pi_{\mathrm{d}}\left(\xi_{\mathrm{d}}\right). \tag{4.1}$$

If we have found this coupling, it is clear that 1 holds. First note that at $\mathrm{t} = 0$, 4.1 is satisfied. Next, suppose that we have finite sets A and B satisfying 4.1, then there is an $\mathrm{x} \in \mathrm{A}$ for each $\mathrm{y} \in \mathrm{B}$. For a death in x, we will say that there is a death in y. If y–1 is not in B, we say there is a birth in y–1 due to the presence of y if there is a birth in any of $(\mathrm{x}_1\text{–}1, \mathrm{x}_2, \ldots, \mathrm{x}_{\mathrm{d}}), \ldots, (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{\mathrm{d}}\text{–}1)$. Due to this construction, the birth rate of $\{\zeta_{\mathrm{t}}\}_{\mathrm{t}\geq 0}$ will be d times the birth rate of $\{\xi_{\mathrm{t}}\}_{\mathrm{t}\geq 0}$ as desired. We do an analogous construction for births in $\mathrm{y}+1$. Any points in $\{\zeta_{\mathrm{t}}\}_{\mathrm{t}\geq 0}$ that are not used will of course generate births independently of $\{\xi_{\mathrm{t}}\}_{\mathrm{t}\geq 0}$. Using this construction, it is clear that 4.1 remains satisfied, since π_{d} maps adjacent points in $\mathbb{Z}^{\mathrm{d}}$ to adjacent points in $\mathbb{Z}$. We have now constructed a coupling with all desired properties, which completes the proof. □

In order to derive an upper bound on $\lambda_{\mathrm{c}}\left(\mathbb{Z}\right)$, we will make a comparison with another stochastic process, called oriented site percolation.

Theorem 4.8. *The process* $\{\xi_{\mathrm{t}}\}_{\mathrm{t}\geq 0}$ *survives if* $\left(1 - \mathrm{e}^{-\lambda \mathrm{T}}\right)\mathrm{e}^{-\mathrm{T}} > \mathrm{p}_{\mathrm{c}}$ *for all* $\mathrm{T} \in \mathbb{R}$. *Here* p_{c} *denotes the critical probability for a certain type of 1-independent oriented percolation.*

Proof. We will use Harris' visual representation of the contact process on the graph $\mathbb{Z}_{\geq 0}$, and make a partition in time with time steps T. We will define an one-independent oriented percolation process based on this visual representation. We let points in $(\mathrm{z}_1, \mathrm{z}_2) \in \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$ correspond to z_1 at time step $\mathrm{z}_2\mathrm{T}$ in Harris's visual representation. We say that the edge $(\mathrm{z}_1, \mathrm{z}_2)\text{–}(\mathrm{z}_1+1, \mathrm{z}_2)$ is open if the following conditions hold:

- There is an infection from z_2 to $\mathrm{z}_2 + 1$ between T and $\mathrm{T}+1$. Denote the time of infection by $\tilde{\mathrm{T}}$.
- There are no recoveries of the node z_1 between $(\mathrm{z}_1 + \mathrm{z}_2)\mathrm{T}$ and $(\mathrm{z}_1 + \mathrm{z}_2)\tilde{\mathrm{T}}$
- There are no recoveries of the node $\mathrm{z}_1 + 1$ between $(\mathrm{z}_1 + \mathrm{z}_2)\tilde{\mathrm{T}}$ and $(\mathrm{z}_1 + \mathrm{z}_2)(\mathrm{T}+1)$

For the vertical edges we say that $(\mathrm{z}_1, \mathrm{z}_2)\text{–}(\mathrm{z}_1, \mathrm{z}_2 + 1)$ is open if

- There are no recoveries of the node z_1 between $(\mathrm{z}_1 + \mathrm{z}_2)\mathrm{T}$ and $(\mathrm{z}_1 + \mathrm{z}_2)(\mathrm{T}+1)$

How paths in the contact process are related to paths in oriented percolation is visualised in Figure 11. Doing this gives an one-independent oriented percolation process where edges are open with probability greater or equal to $\mathrm{e}^{-\mathrm{T}}\left(1 - \mathrm{e}^{-\lambda \mathrm{T}}\right)$. The probabilities are also 1-independent, since the probabilities of an edges being open only depend on whether or not the neighbouring node is open.

This final part of the proof is showing that the even of an infinite path in the oriented percolation process stochastically dominates that of the event of an infinite path in the visual representation which implies that the infection survives.

Let us consider an infinite path in the oriented percolation process, which starts at $(0, 0)$ without loss of generality. If $(0, 0)$ is open there is an infection to node 1, and no recoveries on node 0 or 1 in the next time step. Meaning there is a path to $(0, 1)$ and to $(1, 0)$ in the visual representation as desired. We can repeat this reasoning to all further nodes in the infinite path. This means that an infinite path in our 1-independent oriented percolation process implies a path in the visual representation. □

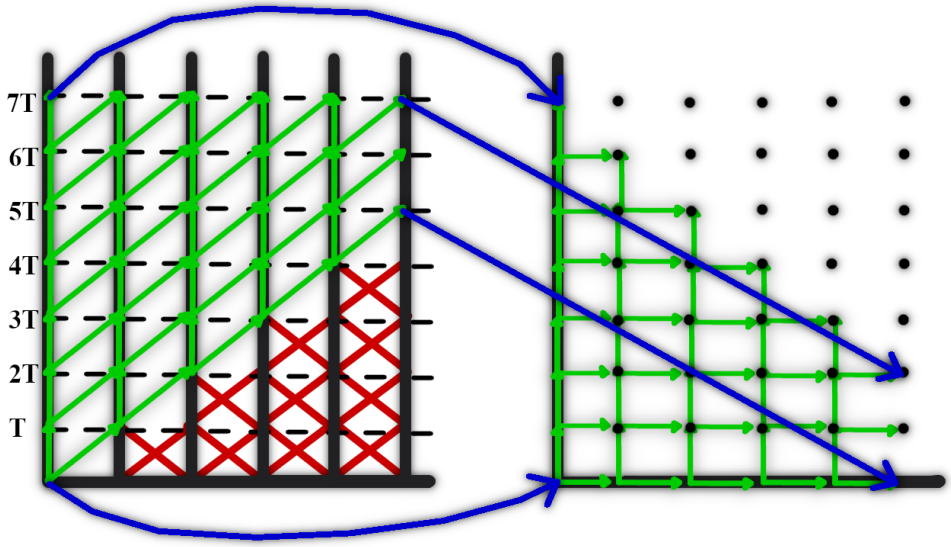


Figure 11: Coupling with Oriented percolation

The event that neighbouring edges are open is positively correlated in the 1-independent percolation model above. This fact suggests that the critical probability of the model is smaller than that of normal oriented percolation.

Lemma 4.9. *The critical value of the* 1*-independent percolation model described in Theorem 4.8 is smaller or equal to* $\mathrm{p_c}(\mathbb{Z}^2)_{oriented}$.

Proof. The proof consists of two steps, we first relate the critical percolation of the 1-independent model to that of another 1-independent percolation model such that the probability of an edge being open, in the absence of other information, is the same for all edges. Secondly, we show that the critical probability of this second model is smaller then that of ordinary oriented percolation.

To do the first step of the proof, note that we can get a 1-independent percolation model where all edges are open with probability $\left(1-\mathrm{e}^{-\lambda \mathrm{T}}\right)\mathrm{e}^{-\mathrm{T}}$ by resampling open vertical edges, so that after the resampling the probability of any vertical edge being open is $\left(1-\mathrm{e}^{-\lambda \mathrm{T}}\right)\mathrm{e}^{-\mathrm{T}}$. Doing this only increases the critical probability, since more edges get closed.

Let Λ_n denote the box with radius n surrounding the origin.

$$\begin{aligned}\mathbb{P}[\underline{0} \leftrightsquigarrow \delta\Lambda_\mathrm{n} \text{in 1-independent model}] &= \mathbb{P}\left[\bigcup_{\text{p path from } \underline{0} \text{ to } \delta\Lambda_\mathrm{n}} \text{p open in 1-independent}\right] \\ &\geq \mathbb{P}\left[\bigcup_{\text{p path from } \underline{0} \text{ to } \delta\Lambda_\mathrm{n}} \text{p open in ordinary OP}\right] \\ &= \mathbb{P}[\underline{0} \leftrightsquigarrow \delta\Lambda_\mathrm{n} \text{in ordinary OP}]\end{aligned}$$

Taking the limit in the inequality above than proves the desired result. □

To complete the upper bound, we refer to one result from the study of oriented percolation. The following result is proven in [1].

Theorem 4.10. *(P. Ballister, B. Bollobàs, A. Stacey, 1993) The critical value of oriented edge percolation is smaller than* 0.6863*.*

Finally we can combine all these results to find an upper bound on the critical value of the contact process on $\mathbb{Z}$, by finding an optimal value of T.

Corollary 4.11. *The value* $\lambda_c(\mathbb{Z})$ *is smaller than* 8.3730*.*

Proof. We have to find the value of T such that

$$\left(1 - e^{-\lambda T}\right) e^{-T} < 0.6863$$

holds for the smallest possible value of $\lambda_{\min}$. Then, Theorem 4.8, Lemma 4.9 and Theorem 4.10 allows us to conclude that $\lambda_c(\mathbb{Z}) < \lambda_{\min}$.

Assuming that T is positive allows us to substitute $T = -\log a$.

$$\begin{aligned}
&\left(1 - e^{\lambda T}\right) e^{-T} = 0.6863 \\
&\left(1 - a^{\lambda}\right) a = 0.6863 \\
&a^{\lambda+1} = a - 0.6863 \\
&\lambda = -1 + \log_a\left(a - 0.6863\right)
\end{aligned}$$

Substituting back in this $a = e^{-T}$ this means we need to minimise $\lambda = -1 + \log_{e^{-T}}\left(e^{-T} - 0.6863\right)$. There exists no nice algebraic expression for this, but numerical estimation gives a reasonable estimate $T = 0.2673$, yielding $\lambda_{\min} < 8.3730$ □

It is worth noting that much stronger connections have been made between the contact process and oriented percolation. In particular the paper by Carol Bezuidenhout and Geoffrey Grimmet [2] needs to be mentioned. In this paper they proved through a much more elaborate coupling then the one here.

To end this chapter we combine all the results on the critical value of the contact process obtained so far.

Corollary 4.12. *The following bounds on the critical value of* $\mathbb{Z}^d$ *hold.*

$$\frac{1}{8} + \frac{\sqrt{18d-1}}{8d\sqrt{2d-1}} \leq \lambda_c(\mathbb{Z}^d) < \frac{8.3730}{d}$$

Proof. Combine Theorem 4.4, Theorem 4.7 and Corollary 4.11. □

These estimates, especially the upper bound, are not particularly tight. The coupling with oriented percolation requires strong results on the critical value of oriented percolation, which makes it a rather weak result. Much tighter bound are available in the literature, namely it is known that [5, p. 308]

$$\frac{1}{2d-1} \leq \lambda_c(\mathbb{Z}^d) < \frac{2}{d}. \tag{4.2}$$

5 Numerical methods

While writing this text, extensive use has been made of computational methods. In this final chapter, I have compiled the most interesting results. Full scripts are provided in the appendix, together with a link to a GitHub repository, for easy access.

GitHub repository

5.1 Visualisation

One of the best tools nowadays for understanding a complex stochastic process, like the contact process, is a good visualisation. Being able to see how the process behaves gives insights that are not obvious from the mathematical definition. In Appendix A.2 one can find the full script tau-leaping simulation of the contact process on a small part of $\mathbb{Z}^2$, plotting how it evolves in real-time. It includes controls for adjust the rate in real time with the scroll wheel, toggling full screen with F11 and toggling a graph with the total number of infections using the left mouse button.

The heart of the simulation is the `contact_process` function. We generate Poisson events for recovery in line 9, and for infections from its neighbours in line 13. The final state of the node is then updated in global list `inf` according to the most recent event.

```
1  def contact_process(graph, dt, rate):
2  '''
3  graph     :   The graph on which we do the contact process
4  dt        :   The simulation timestep
5  rate      :   The rate of the contact process
6  Updates the inf array to represent the contact process and append the number of
   ↪  infection at this time to inf_count
7  '''
8  for i in graph.nodes():
9      recov_time = generate_poisson_events(1/1000, dt)
10     inf_time = -1
11     for j in nx.all_neighbors(graph, i):
12         if inf[j] == 1:
13             inf_time = max(inf_time, generate_poisson_events(rate/1000, dt))
14     if inf_time >= 0 or recov_time >= 0:
15         inf[i] = 0 if recov_time > inf_time else 1
16 inf_count.append(sum(1 for value in inf.values() if value == 1))
```

One might criticise this simulation for using a tau-leaping approach, as it is slightly inaccurate. However, this approach is faster when a large number of nodes is infected. This makes it ideal visualisation, as drawing the picture already takes up a large amount of the computing resources.

5.2 Improved lower bound

In Lemma 4.3 we derived a result which allows us to find a lower bound on the critical value of the contact process. We have done so for 2 frontier events in Theorem 4.4. It is interesting to ask ourselves, how far can we go? In the subsection we will first numerically prove a stronger lower bound in one dimension, using the computer algebra package `sympy`. Finally, we will numerically calculate the expected value for $\lambda = 1.05$ to show that this value is a lower bound of the critical value $\lambda_c(\mathbb{Z})$.

Let us first look at the algebraic solver. We will consider what the contact process looks like, with an equivalence up to horizontal shifts. That means that in our program we will consider

$\xi_t = \{-1, 1, 2\}$ to be the same as $\xi_t = \{2, 4, 5\}$. This will drastically reduce the number of steps our algorithm takes as we consider larger numbers of frontier events.

In our program we will start with a dictionary containing the state string `xox` and associated probability 1. Now, to calculate the distribution of frontiers after the next frontier event, we loop through every state string. For this state string we will consider all possible events and the probability that they occur. We track of these states and probabilities in a new dictionary, which can than be used for the next iteration of the loop.

A full implementation can be found in Appendix A.3. Using this implementation we now find the next theorem.

Theorem 5.3. *The following lower bound holds*

$$\lambda_c(\mathbb{Z}) \geq 0.8958876.$$

Proof. Using the script in Appendix 4.3 until $d = 11$ together with Lemma 4.4 yields that $\lambda_c(\mathbb{Z}) \geq 0.8958876$. If we plot the expected value to the rate λ we see □

Is this kind of computer proof valid?

However we can go further still. If we fix the value of λ and omit the symbolic expression the script can be sped up significantly, allowing us to go to greater depths than $d = 11$. We see that for $\lambda = 1$ the expected value dips below 1 after $d = 24$. However, not unsurprisingly, once we increase the value to $\lambda = 1.1$, by $d = 24$ the expected value is still approximately 1.29.

Theorem 5.4. *The following lower bound holds,*

$$\lambda_c(\mathbb{Z}) \geq 1.05.$$

Proof. Using the adapted script found in Appendix A.3 we see that with rate 1.05, for $d = 34$ the expected number of children after d frontier events is approximately 0.999052027980148 and hence with great certainty smaller than 1. Allowing us to conclude the lower bound with Lemma 4.4 □

For both of the previous theorems, results were obtained after less than an hour of computation time. The amount of possible states after d frontier events scales roughly like 2^d, so the complexity of the algorithm is exponential. Given this fact, its unlikely that more computing time will lead to drastically better results.

5.5 Numerical estimation of the critical value

Finally, we want to see if we can numerically determine the critical value of the contact process on $\mathbb{Z}$. We will use the bisection algorithm to find the right most zero of the survival function ψ as defined in Definition 2.10. To do so, we will employ a suitable Monte Carlo method. We will simulate all the events in a contact process, starting with just the origin infected, for some number of attempts. If in any of these attempts, we exceed the threshold, this value is right of the zero. Otherwise we assume it to be left of the rightmost zero.

The simulation stores the contact process in a dictionary with tuples as keys storing node class objects. This node class contains a boolean to indicate whether the node is infected, and a count of all the uninfected neighbours of this node. Indexing our dictionary with tuples allows us to write the algorithm for arbitrary dimensions.

To generate the next event, we uniformly sample in $[0, 1]$ and than pick whether or not this event is a recovery or an infection by checking if our sampled value is smaller than the total infection rate divided by the total infection and recovery rate, than we uniformly sample an

valid edge to infect or node to recover respectively. This is a valid approach by Lemma 1.13, and is much faster than simulating Poisson processes.

An implementation of this method is included in Appendix A.4. Some steps have been taken to optimise this script, for example, we are using the `operator.add` function to utilise `c` language speeds for checking all neighbouring tuples. Additionally, we use the `loky` library to allow for multiprocessing, yielding practical speed-up of about 4 to 8 times.

Using a threshold of 1000 infections, with 25 attempts, we tried estimating the critical value of the contact process for dimensions 1 to 20, starting from the bounds by Liggett 4.2. The threshold was too low to obtain any meaningful results for dimension 11 to 20, but for dimension 1 to 10 we got the following results after several hours of computing.

Dimension	1	2	3	4	5	6	7	8	9	10
Estimate	1.646	0.413	0.219	0.153	0.114	0.099	0.079	0.072	0.060	0.054

Table 1: Estimates of the critical value for dimension 1 to 10

One might wonder how good these estimates are. However, we can compare them to other numerical estimates, like the ones by Munir Sabag and Mário de Oliveira [7], who did estimations up to dimension 5 for the conserved contact process back in 2002. It is noted by the authors that these values seems to agree with those of the ordinary contact process, which suffices for our purpose. Their estimations, corrected to undo normalisation, are shown in Table 2.

Dimension	1	2	3	4	5
Estimate	1.6489	0.41218	0.21947	0.14938	0.11384

Table 2: Estimates of the critical value for dimension 1 to 5 by Sabad and de Oliveira

Comparing these values with our estimations, we see that for these first 5 dimensions, we have a deviation of at most 0.005.

We can also employ this method to estimate how our survival function ψ from Definition 2.10 develops by counting the number of times we exceed the threshold in our total amount of attempts. Since this requires us to do many more attempts, we have to use a lower threshold to stay within a reasonable amount of computing time. One of these estimations was shows in Figure 4. This was generated with 250 attempts and a threshold of 100, requiring about 2 hours of computing. We calculated an empirical probability for several values, increasing the density as we approach the critical value, to compensate for the larger slope.

6 References

[1] Paul Balister, Bela Bollobas, and Alan Stacey. Upper bounds for the critical probability of oriented percolation in two dimensions. *Proceedings: Mathematical and Physical Sciences*, 440(1908):201–220, 1993.

[2] Carol Bezuidenhout and Geoffrey Grimmett. The Critical Contact Process Dies Out. *The Annals of Probability*, 18(4):1462 – 1482, 1990.

[3] Geoffrey Grimmett. *Percolation*, volume 321 of *Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, second edition, 1999.

[4] G.F. Lawler. *Introduction to Stochastic Processes*. Chapman & Hall/CRC Probability Series. CRC Press, 2018.

[5] T.M. Liggett. *Interacting Particle Systems*. Grundlehren der mathematischen Wissenschaften : a series of comprehensive studies in mathematics. Springer-Verlag, 1985.

[6] S.M. Ross. *Introduction to Probability Models*. Probability and Statistics. Academic Press, 2007.

[7] Munir M. S. Sabag and Mário J. de Oliveira. Conserved contact process in one to five dimensions. *Phys. Rev. E*, 66:036115, Sep 2002.

[8] R. van der Hofstad. *Random Graphs and Complex Networks*. Number v. 1 in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2017.

A Python scripts

All scripts have been confirmed to work with Python version 3.13.2. In the table below you can find all packages used, with the specific version.

Package	Version	Available at
numpy	2.1.3	`https://pypi.org/project/numpy/2.1.3/`
matplotlib	3.10.1	`https://pypi.org/project/matplotlib/3.10.1/`
scipy	1.15.2	`https://pypi.org/project/scipy/1.15.2/`
networkx	3.4.2	`https://pypi.org/project/networkx/3.4.2/`
loky	3.5.2	`https://pypi.org/project/loky/3.5.2/`
psutil	7.0.0	`https://pypi.org/project/psutil/7.0.0/`

The code in the chapter has been reduced to a minimal working example, wherever that makes sense. This means that all code that is not relevant to solving the problem at hand, such as plot labels, has been removed.

A.1 Expected extinction time on cycle graphs

In subsection 3.1 we derived a system of equation to find and lower and upper bound on the expected extinction time of the contact process on a cyclic graph given different initial configurations. We used a symbolic solver to derive Figure 8. This appendix contains the scripts to solve these systems. First, we have a script to solve the system for the lower bound algebraically.

```
1  from sympy import symbols, Rational
2  import sympy.plotting as spp
3  from sympy.matrices import Matrix
4  import math
5  '''
6  We define a sympy symbol, specify the dimension of the system, and initialise the
   ↪  matrix and vector.
7  '''
8  l = symbols("l")
9  n = 5
10 b = [1/(2*l + i + 1) for i in range(n - 1)] + [Rational(1, n)]
11 A = []
12 for i in range(n - 1):
13     A.append([-(i + 1)/(2*l + i + 1) if i == j + 1 else 1 if i == j else -(2 *
       ↪  l)/(2*l + i + 1) if i == j - 1 else 0 for j in range(n)])
14 A.append([0 for _ in range(n-2)] + [-1, 1])
15 '''
16 We solve the system, and plot the result.
17 '''
18 r = Matrix(A).LUsolve(Matrix(b))
19 p1 = spp.plot(r[-1], (l, 0, 3), show=False)
20 p1.show()
```

We can easily adapt this script to solve the upper bound system, by replacing line 10 to line 14 with the following code snippet.

```
b = [1 / (2 * i * l + i + 1) if i <= math.floor((n - 1) / 2) else 1 / (2 * (n - i - 1) * l + i + 1) for i in range(n - 1)] + [Rational(1,n)]
A = []
for i in range(n - 1):
    A.append([
        -(i + 1) / (2 * min(i, n - i - 1) * l + i + 1) if i == j + 1 else 1 if i == j else
        -(2 * min(i, n - i - 1) * l) / (2 * min(i, n - i - 1) * l + i + 1) if i == j - 1 else 0 for j in range(n)
    ])
A.append([0 for _ in range(n-2)] + [-1, 1])
```

We also studied the behaviour of the bounds for fixed lambda, as n grows. To solve these larger systems, we use a sparse matrix implementation. This can solve the systems at a fraction of the time required compared to the regular solver. These scripts have been used to generate the plots in Figure 9.

```
from scipy.sparse import diags
from scipy.sparse.linalg import spsolve
import math
import numpy as np
import matplotlib.pyplot as plt
'''
We define a rate, specify the range of dimension for which we solve the system, and initialise the matrix and vector.
'''
l = 0.25
n_values = range(2, 40)  # Range of n values to observe growth
solutions = []
'''
Solve the system and plot the solution
'''
for n in n_values:
    b = np.array([1 / (2 * l + i + 1) for i in range(n - 1)] + [1 / n])
    diagonals = [
        [-((i + 1) / (2 * l + i + 1)) for i in range(1, n)],  # Sub-diagonal
        [1] * n,                                              # Main diagonal
        [-(2 * l) / (2 * l + i + 1) for i in range(n - 1)]    # Super-diagonal
    ]
    A = diags(diagonals, offsets=[-1, 0, 1], shape=(n, n)).toarray()
    A[-1, -2:] = [-1, 1]
    r = spsolve(A, b)
    solutions.append(r[-1])
plt.plot(n_values, solutions)
plt.show()
```

Where again the script can easily be adapt to work for the upper bound by replacing line 15 to 19 by the next snippet.

```
b = np.array([1 / (2 * i * l + i + 1) if i <= math.floor((n - 1) / 2) else 1 / (2 * (n - i - 1) * l + i + 1) for i in range(n - 1)] + [1 / n])
diagonals = [
```

```
    [-(i + 1) / (2 * min(i, n - i - 1) * l + i + 1) for i in range(1, n)],  # Sub-diagonal
    [1] * n,  # Main diagonal
    [-(2 * min(i, n - i - 1) * l) / (2 * min(i, n - i - 1) * l + i + 1) for i in range(n - 1)]  # Super-diagonal
]
```

A.2 Visualisation

The following script does an estimation of the contact process as described in subsection 5.1. As a reminder, it includes controls to adjust the rate in real time with the scroll wheel, toggle full screen with F11 and toggle a graph with the total number of infections using the left mouse button.

```
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import tkinter as tk
from tkinter import *
import psutil
import numpy as np
import networkx as nx

'''
We configure a Tkinter window, and the associated objects required to draw the network and graph.
'''
root = Tk()
defaultFont = tk.font.nametofont('TkDefaultFont')
frame= Frame(root).pack(fill= BOTH, padx= 0, pady=0)
root.title("Contact process")
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
fig = plt.figure(frameon=True, figsize=(screen_width/100,screen_height/100), dpi=102)
canvas = FigureCanvasTkAgg(fig, root)

'''
Keybinds:
button_0    :   Display a graph that shows the number of infection over time.
scroll      :   Increase/decrease the rate of infection.
F11         :   Toggle fullscreen.
'''
root.bind("<F11>", lambda e: root.attributes("-fullscreen", not root.attributes("-fullscreen")))
root.bind("<Button-1>", lambda event: globals().__setitem__('display_infections', not display_infections))
root.bind("<MouseWheel>", lambda event: globals().__setitem__('r', max(0, r + 0.1 if event.delta > 0 else r - 0.1)))

'''
Main variables:
```

```
G                   :   The graph on which we run the contact process.
dt                  :   The timestep in ms at which we update the graph.
T                   :   The final time in ms the simulation
r                   :   The initial infection rate per second
display_infection   :   Bool that determines whether or not a graph is displayed that shows recent infections.
inf                 :   A list of infected nodes
inf_count           :   Stores the number of infected nodes at each timestep
'''
scale = 35
G = nx.grid_2d_graph((int)(screen_height/scale), (int)(screen_width/scale))
dt, T, r = 500, 1000000, 0.5
global display_infections; display_infections = False
inf = {list(G)[i] : 1 for i in range(len(list(G)))}
pos = {(x,y):(y,-x) for x,y in G.nodes()}
inf_count = []

def generate_poisson_events(rate, time_duration):
    '''
    rate            :   The rate of the Poisson process
    time_duration   :   The time interval in which to generate events.
    Runs the poisson process and returns the time of the last hit. If not hits occur, returns -1.
    '''
    num_events = np.random.poisson(rate * time_duration)
    event_times = np.sort(np.random.uniform(0, time_duration, num_events))
    if num_events > 0:
       return np.max(event_times)
    return -1

def contact_process(graph, dt, rate):
    '''
    graph     :   The graph on which we do the contact process
    dt        :   The simulation timestep
    rate      :   The rate of the contact process
    Updates the inf array to represent the contact process and append the number of infection at this time to inf_count
    '''
    for i in graph.nodes():
        recov_time = generate_poisson_events(1/1000, dt)
        inf_time = -1
        for j in nx.all_neighbors(graph, i):
            if inf[j] == 1:
                inf_time = max(inf_time, generate_poisson_events(rate/1000, dt))
        if inf_time >= 0 or recov_time >= 0:
            inf[i] = 0 if recov_time > inf_time else 1
    inf_count.append(sum(1 for value in inf.values() if value == 1))

'''Generates colours for networkx to draw infected nodes red, and healthy ones blue'''
def generate_colour_map(inf):
    return ['#BF211E' if value == 1 else '#1f77b4' for value in inf.values()]

```

```
82
83  def timeStep(current_time):
84      '''
85      current_time    :   The time at which we want to display the contact process
86      This function clears the previous figure does the following three things
87      1.  Draw a label that show the current rate
88      2.  Draw G
89      3.  If display_infections is true, draw a graph that shows the number of
        ↪  infections
90      '''
91      plt.clf()
92      r_label = Label(root, text=f"Rate (l): {r:.2f}", font=("Open Sans", 36),
        ↪  fg="black", bg="white", bd=5, relief="solid")
93      r_label.place(relx=0.5, rely=0.03, anchor="n")
94      r_label.config(text=f"Rate (l): {r:.2f}")
95
96      col = generate_colour_map(inf)
97      ax = plt.gca()
98      ax.set_xlim([min(x for x, y in pos.values()) + 1, max(x for x, y in
        ↪  pos.values()) - 1])
99      ax.set_ylim([min(y for x, y in pos.values()) + 1, max(y for x, y in
        ↪  pos.values()) - 1])
100     ax.set_axis_off()
101     nx.draw(G, pos, ax=ax, node_size=screen_width/(0.1*scale), node_color=col,
        ↪  with_labels=False, edgecolors='black', linewidths=1.5, node_shape='s')
102     plt.subplots_adjust(left=0, right=1, top=1, bottom=0)
103
104     if(display_infections):
105         plt.subplot(1, 2, 1, facecolor=(1, 1, 1, 0.8))
106         plt.plot(inf_count[max(0, len(inf_count) - 100): len(inf_count) -
            ↪  1],linewidth=5)
107         plt.ylim(0, len(G.nodes()))
108         plt.xlabel("time (ms)", labelpad=-35, loc='center', fontsize=18),
            ↪  plt.ylabel("infections", labelpad=-35, loc='center', fontsize=18)
109         plt.title("Infected sites over time", pad=-20, loc='center', y=0.95,
            ↪  fontsize=36)
110         plt.gca().tick_params(labelsize=0)
111         for spine in plt.gca().spines.values():
112             spine.set_edgecolor('black')
113             spine.set_linewidth(5)
114
115     canvas.draw()
116     canvas.get_tk_widget().pack(side=TOP, fill=BOTH, expand=True)
117
118 '''
119 Main loop:
120 Runs the contact and visualises the contact process until time T.
121 '''
122 for i in range((int)(T/dt)):
123     if not (1 in inf.values()):
124         break
125     root.update_idletasks()
126     root.update()
```

```
    contact_process(G, dt, r)
    root.after(0, timeStep(i*dt))
root.mainloop()
```

A.3 Improved lower bound

This is the script associated with the algebraic solver in subsection 5.2.

```
import sympy as sp
'''
First initialise key variables.  The variable initial is a dictionary that has all frontiers in a step,
with the associated probability. In this dictionary, zeroes denote infected sites, crosses healthy ones.
The max number of iterations the program will run is stored in d. For the sympy algebra we define a
symbol lambda.
'''
initial = { "xox" : 1}
d = 10
l = sp.Symbol('l')
'''
state   :   String that represents a frontier state.
Finds total rate of events that change the frontier.
'''
def getTotalRate(state):
    rate = 2*l
    for i in range(1, len(state) - 1):
        rate += 1 if state[i] == 'o' else l * (int(state[i - 1] == 'o') + int(state[i + 1] == 'o'))
    return rate
'''
string      :   String to change
insert      :   The value to insert into string
position    :   Position at which we will insert string
Returns string with insert inserted at position.
'''
def stringReplace(string, insert, position):
    if position == 0:
        return insert + string[1:]
    elif position == len(string) - 1:
        return string[:len(string) - 1] + insert
    return string[:position] + insert + string[position + 1:]
'''
heal        :   A bool that indicates whether or not the event is a recovery
i           :   The current index in state
totalRate   :   The total rate of infection/healing in state
state       :   A string that indicates currect state of the frontier
probability :   The probability of reaching state
next        :   The dictionary of possible states with probabilities after one more event
Returns updated next with the events at i accounted for.
'''
```

```
def event(heal, i, totalRate, state, probability, next):
    if(heal):
        st = stringReplace(state, 'x', i)
        while st[1] == 'x' and st != 'xx':
            st = st[1:]
        while st[len(st) - 2] == 'x' and st != 'xx':
            st = st[:len(st) - 1]
        prob = probability * 1/totalRate
    else:
        st = stringReplace(state, 'xo' if i == 0 else 'ox' if i == len(state) - 1 else 'o', i)
        prob = probability * 1/totalRate * l
    next[st] = next.get(st, 0) + prob
    return next
'''
state_dict  :   Dictionary with frontier states and associated probabilities
Returns the expected number of children with the frontier distribution state_dict.
'''
def expectation(state_dict):
    return sum(prob * state.count('o') for state, prob in state_dict.items())

'''
Using the event function, this snippet of code determine and prints the expect number
of infections after d frontier events. We loop through all states, and to all nodes
in these states, appending the result of every possible event with the associated
probability to next.
'''
next_state = initial
for index in range(d+1):
    next = {}
    for state in next_state:
        probability = next_state[state]
        totalRate = getTotalRate(state)
        if(state == "xx" ):
            if (state in next):
                next[state] += probability
            else:
                next.update({ state :  probability} )
        elif(state != "xx" ):
            next = event(False, 0, totalRate, state, probability, next)
            next = event(False, len(state) - 1, totalRate, state, probability, next)
            for i in range(1, len(state) - 1):
                if state[i] == 'o':
                    next = event(True, i, totalRate, state, probability, next)
                else:
                    if state[i - 1] == 'o':
                        next = event(False, i, totalRate, state, probability, next)
                    if state[i + 1] == 'o':
                        next = event(False, i, totalRate, state, probability, next)
    next_state = next
    print(f"iteration {index}: l = {sp.nsolve(expectation(next_state) - 1, l, 1)}")
'''
```

```
92 We finally plot the solution to show how the expected value develops.
93 '''
94 p1 = sp.plot(expectation(next_state), xlim = (0, 3), ylim = (0,2), show=False)
95 p1.show()
```

We can do some slight adaptations to the script to do numerical approximations of the lower bound. By changing `l` in line 10 to a fixed value an than replacing line 61 to 96 with the following snippet we get the expected value for a fixed value of λ.

```
'''
Using the event function, this snippet of code determine and prints the expect
↪  number
of infections after d frontier events. We loop through all states, and to all nodes
in these states, appending the result of every possible event with the associated
probability to next.
'''
next_state = initial
for index in range(d):
    next = {}
    for state in next_state:
        probability = next_state[state]
        totalRate = getTotalRate(state)

        if(state == "xx"):
            if (state in next):
                next[state] += probability
            else:
                next.update({ state :  probability} )

        elif(state != "xx"):
            next = event(False, 0, totalRate, state, probability, next)
            next = event(False, len(state) - 1, totalRate, state, probability,
            ↪  next)

            for i in range(1, len(state) - 1):
                if state[i] == 'x':
                    if state[i - 1] == 'o':
                        next = event(False, i, totalRate, state, probability, next)
                    if state[i + 1] == 'o':
                        next = event(False, i, totalRate, state, probability, next)
                elif state[i] == 'o':
                    next = event(True, i, totalRate, state, probability, next)
    next_state = next
    print("expected children at depth " + str(index) + " = " +
    ↪  str(expectation(next_state)))
```

A.4 Numerical estimation of the critical value

This final script numerically estimates the critical value of the contact process for the dimensions 1 to 20. The script can easily be modified to exclusively generate estimates for dimension 1 or to estimate the survival functions ψ.

```
import numpy as np
import loky
from loky import as_completed
import random
import operator
'''
attemps          :      The total number of contact processes that are tried.
thres            :      The total number of infections required to declare survival
max_dimension    :      The largest dimension for which we will determine the
↪  critical probability.
'''

attempts = 25
thres = 500
max_dimension = 20
max_steps = np.inf
neighbour_offset = []
'''
Node object to represent each node in the contact process
Each node has an infected status and a count of uninfected neighbours
'''
class Node:
    def __init__(self, infected = True, uninfected_neighbours=2):
        self.infected = infected
        self.uninfected_neighbours = uninfected_neighbours
    def __repr__(self):
        return f"Node(infected={self.infected},
        ↪  uninfected_neighbours={self.uninfected_neighbours})"
'''
Runs one timestep of the contact process
total_infections    :    Total number of infections in the system
total_rate          :    Total rate of infection in the system
key_list            :    List of keys representing the current state of the system
contact_process     :    Dictionary representing the current state of the system
rate                :    The rate of infection
Returns the updated list of keys, contact process, total infections, and total rate
'''
def time_step(total_infections, total_rate, key_list, contact_process, rate):
    infection = np.random.uniform(0, 1) < total_rate / (total_infections +
    ↪  total_rate)
    if(not infection):
        index = np.random.randint(0, len(key_list))
        toheal = key_list[index]
        contact_process[toheal].infected = False
        total_rate -= contact_process[key_list[index]].uninfected_neighbours * rate
        total_infections -= 1
        for offset in neighbour_offset:
            neighbour_key = tuple(map(operator.add, key_list[index], offset))
            contact_process[neighbour_key].uninfected_neighbours += 1
            if contact_process[neighbour_key].infected:
                total_rate += rate
        key_list[index] = key_list[-1]
        key_list.pop()
    elif(infection):
```

```
        options = []
        infect = random.choices(key_list,
        ↪  weights=[contact_process[key].uninfected_neighbours for key in
        ↪  key_list], k=1)[0]
        for offset in neighbour_offset:
            neighbour_key = tuple(map(operator.add, infect, offset))
            if neighbour_key in contact_process and not
            ↪  contact_process[neighbour_key].infected:
                options.append(neighbour_key)

        new = np.random.randint(0,len(options))
        contact_process[options[new]].infected = True
        total_rate += contact_process[options[new]].uninfected_neighbours * rate
        total_infections += 1
        key_list.append(options[new])
        for offset in neighbour_offset:
            neighbour_key = tuple(map(operator.add, options[new], offset))
            if neighbour_key not in contact_process:
                contact_process[neighbour_key] = Node(False, 2*dimension - 1)
            else:
                contact_process[neighbour_key].uninfected_neighbours -= 1
                if contact_process[neighbour_key].infected:
                    total_rate -= rate
    return total_infections, total_rate, key_list, contact_process

'''
Threshold   :    The threshold for the number of infections
dimension   :    Dimension of the graph Z^d
Run one iteration of the contact process until the number of infections is greater
↪  than 100 or smaller or equal to 0
Returns True if the number of infections is greater than or equal to the threshold,
↪  False otherwise
'''
ti = 0
tr = 0
def run_contact_process(rate, dimension):
    akeys_lst = []
    akeys_lst.append(tuple((0 for _ in range(dimension))))
    cprocess = {tuple([0 for _ in range(dimension)]): Node(True, 2*dimension)}
    for i in range(dimension):
        for offset in [1, -1]:
            neighbour_key = tuple(tuple((0 for _ in range(dimension)))[j] + (offset
            ↪  if j == i else 0) for j in range(dimension))
            cprocess[neighbour_key] = Node(False, 2*dimension - 1)
    ti = 1
    tr = 2 * rate * dimension
    steps = 0
    while ti < thres and ti > 0 and steps < max_steps:
        steps += 1
        ti, tr, akeys_lst, cprocess = time_step(ti,tr,akeys_lst, cprocess, rate)
    if ti >= thres:
        return True
    return False
```

```
99
100 ''''
101 low          :    The lower bound for the critical value
102 high         :    The upper bound for the critical value
103 attempts     :    The number of attempts to run the contact process
104 epsilon      :    The tolerance for the bisection method
105 dimension    :    Dimension of the graph Z^d
106 Given a lower and upper bound, estimate the critical value using bisection method.
107 Returns lower and upper estimate for the critical value
108 '''
109 def crit_value_bisection(low, high, attempts, epsilon, dimension):
110     with loky.get_reusable_executor(max_workers=14) as executor:
111         while high - low > epsilon:
112             found_true = False
113             tasks_completed = 0
114             mid = (low + high) / 2
115             futures = set()
116             for _ in range(attempts):
117                 future = executor.submit(run_contact_process, mid, dimension)
118                 futures.add(future)
119             for future in as_completed(futures):
120                 tasks_completed += 1
121                 result = future.result()
122                 if result:
123                     found_true = True
124                 if found_true:
125                     break
126             if found_true:
127                 high = mid
128             else:
129                 low = mid
130             print(f"The critical value is likely between {low:.6f} and
                ↪  {high:.6f}.")
131     return high, low
132
133 '''
134 Determine the critical value in for various dimensions.
135 '''
136 for dimension in range(1, max_dimension + 1):
137     neighbour_offset = []
138     for i in range(dimension):
139         for offset in [1, -1]:
140             neighbour_offset.append([offset if j == i else 0 for j in
                ↪  range(dimension)])
141     rate_low = 1/(2*dimension - 1)
142     rate_high = 2/(dimension)
143     rate_high, rate_low = crit_value_bisection(rate_low, rate_high, attempts,
    ↪  0.001, dimension)
144     print(f"The critical value in dimension {dimension} is likely between
    ↪  {rate_low:.6f} and {rate_high:.6f}.")
```