Network Decontamination with Temporal Immunity

by

Yassine Daadaa

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Network decontamination is a well known mobile agent problem with many applications. We assume that all nodes of a network are contaminated (e.g., by a virus) and a set of agents is deployed to decontaminate them. An agent passing by a node decontaminates it, however a decontaminated node can be recontaminated if any of its neighbours is contaminated. In the vast literature a variety of models are considered and different assumptions are made on the power of the agents.

In this thesis we study variation of the decontamination problem in mesh and tori topologies, under the assumption that when a node is decontaminated, it is immune to recontamination for a predefined amount of time t (called *immunity time*). After the immunity time is elapsed, recontamination can occur.

We focus on three different models: mobile agents (MA), cellular automata (CA), and mobile cellular automata (MCA). The first two models are commonly studied and employed in several other contexts, the third model is introduced in this thesis for the first time. In each model we study the temporal decontamination problem (adapted to the particular setting) under a variety of assumptions on the capabilities of the decontaminating elements (agents for MA and MCA, decontaminating cells for CA). Some of the parameters we consider in this study are: visibility of the active elements, their ability to make copies of themselves, their ability to communicate, and the possibility to remember their past actions (memory). We describe several solutions in the various scenarios and we analyze their complexity. Efficiency is evaluated slightly differently in each model, but essentially the effort is in the minimization of the number of simultaneous decontaminating elements active in the system while performing the decontamination with a given immunity time.

Acknowledgements

First and above all, I would like to express my gratitude to Allah for providing me the blessings to complete this work. This thesis appears in its current form due to the assistance and guidance of several people. I would therefore like to offer my sincere thanks to all of them.

I would like to express my thanks to my advisors, Prof. Nejib Zaguia and Prof. Paola Flocchini. I am specially thankful to Prof. Zaguia for his continued encouragement and invaluable suggestions during this work. I thank his family members for their kindness and affection, and for never letting me feel that I am away from my parents and country. I am very thankful to Prof. Paola Flocchini for her supervision, advice, and guidance from the very early stage of this research as well as giving me extraordinary experiences throughout the work. Above all and the most needed, she provided me unflinching encouragement and support in various ways. Her truly scientific intuition has made her a constant oasis of ideas and passions in science, which exceptionally inspire and enrich my growth as a student, a researcher. She is the kind of scientist that I aspire to be. scientist want to be. I always found her in times of need since my first year of undergraduate studies in Computer Science. I am indebted to her more than she knows.

I gratefully thank Prof. Nicolas Santoro, Prof. Amiya Nayak, Prof. Guy Vincent Jourdan, and Prof. Lucien Haddad for their constructive comments on this thesis. I am thankful that in the midst of all their activity, they agreed to be examiners for this thesis.

I dedicate this thesis to my family, my father Mohamed Taher, my mother Aziza, my sisters Mouna, Salsabille, and Sana, my brothers Sofiene and Ahmed, my grandfather Said, my aunt Aicha, and my uncles Ahmed, Hachemi, Belgacem, Taher, and Mohamed for their constant support, love and motivation.

My special thanks go to Eya, Arij, Amani, Myriam, Radhia, Mouna, Anis, Dhia, Billel, Abdallah, Mohamed, Yassine, Nizar, Ahmed, Hamdi, Sayed, Taha, Khaled, Taoufik, Basset, Walid, Karim, Adnen, Mohamed Salah, Sami, and Saber for their love and support.

Finally, I would like to thank everybody who was important to the successful realization of this thesis, as well as expressing my apology that I could not mention personally one by one.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Faults and viruses often spread in networked environments where nodes represent hosts and edges represent connections between hosts, by propagating from site to neighbouring site. The process is called *network contamination*. Once contaminated, a node might behave incorrectly, and it could cause its neighbouring cells to become contaminated as well, thus propagating faulty computations. The propagation patterns of faults can follow different dynamics, depending on the behaviour of the affected site. An obvious and pressing concern for fault tolerance and security is to devise strategies to correct the faulty behaviour at network sites and to neutralize the propagation of faults (*decontamination*). The thesis is concerned with the decontamination issue in cellular systems.

1.1 Problem Statement

A two-dimensional cellular space (i.e., a grid) is considered to be initially fully contaminated by a virus (or completely faulty). The system evolves synchronously and the problem is to distributively decontaminate the space to reach a moment when all cells are simultaneously clean. The decontamination strategy to be devised depends on the dynamics of the spread of faults as well as on the model employed for the actual decontamination process. An agent is able to decontaminate any infected node it visits; however, once it departs, the decontaminated node can be recontaminated by infected neighbours. We say that the network has time immunity if once the agent departs, the decontaminated node is immune for t time units to viral attacks from infected neighbours. After the immunity time is elapsed, recontamination can occur. The value t is also called immunity time.

The various dynamics of *contamination* range from *full spread* to *selective spread*. A full spread behaviour means that when a site is affected by a virus or any other malfunction, such a malfunction can propagate to all its neighbours (e.g., see [92, 5, 47, 57]). When the spread is selective, on the other hand, faults only propagate to sites that are susceptible. The definition of susceptibility depends on the application but often it is based on local conditions. For example, a node could be vulnerable to contamination if the majority of its neighbours are faulty, immune otherwise (e.g., see [88, 89, 93]), or a node could be immune to contamination for a certain amount of time after being repaired (e.g., see [56, 26]).

Among the various possible techniques of *decontamination*, two types have been identified in the literature (for a survey see [46] and for more details see Chapter 2): internal and external decontamination.

In *internal decontamination* a site can decontaminate itself (i.e., it can activate an antiviral software) when a certain condition of the neighbourhood is verified. A clean site gets recontaminated when some other condition of the neighbouring states is verified. This approach has been followed in [102], where a node becomes clean when the majority of its neighbours are clean, and a clean node becomes contaminated if any of its neighbours is contaminated. A similar approach has been taken in [92, 93], where a decontaminated node is immune to recontamination if the majority of its neighbours is clean. Internal decontamination has been specifically studied in the context of fault-tolerance to describe the mechanisms of the spread of faults and of auto-correction (see [103] for a survey). In all these studies the main objective is typically to determine the minimum size of a set of faulty nodes which completely disrupts the system under given contamination/decontamination dynamics or, equivalently, the minimum size of a set of decontaminating nodes that can decontaminate the whole network under the same circumstances.

On the other hand, *external decontamination* is performed by *mobile agents* moving in the network. There is an extensive literature on external decontamination either in specific topologies (see [5, 47]), under various assumption on the capabilities of the agents, or in arbitrary topologies (see [14]). Typically, agents have memory, distinct identifiers, the ability to communicate with other agents when they meet or exchange information writing on whiteboards (storage area located at the nodes). In all models investigated, agents can move from node to node (usually asynchronously and independently) decontaminating the sites they pass through. A clean site becomes contaminated if at least one of its neighbours is contaminated. External decontamination has been studied in the

context of intruder capture to design algorithms to neutralize a virus in a network, or in graph search (e.g., [5, 62, 47, 14, 7]). The main goal of decontamination in these settings is usually to design a strategy that employs the minimum possible size of the team of cleaning agents.

In this thesis we consider a *selective spread* of contamination based on what we call *temporal immunity*: a clean node is allowed to stay in contact with a contaminated node for a predefined amount of time (the *immunity time*) after which it becomes *contaminated*. Based on this definition of contamination we consider three different models for performing decontamination: *Cellular Automata*, *Mobile Agents*, and *Mobile Cellular Automata*. The first two models are commonly studied and employed in other contexts, the third has been introduced in this thesis for the first time. Cellular automata provide a form of internal decontamination, while mobile agents provide a form of external decontamination. Mobile cellular automata are introduced here to show a hybrid decontamination which is neither internal nor external and can be seen as an intermediate model between the other two. In mobile cellular automata, in fact, we have the presence of *active entities* (like in the mobile agents model) which move around to decontaminate nodes they pass by, as well as *locality* (as in cellular automata) which forces the active entities to move on the basis of knowledge of their neighbourhood.

1.2 Objective

The main objectives of the thesis are to provide efficient decontamination strategies in each of the three models, and to make the first steps towards the understanding of their computational power in the context of network decontamination. Efficiency will be evaluated slightly differently in each model, but essentially we will try to minimize in each model the *number of simultaneous decontaminating elements* active in the system while performing the decontamination with a certain immunity time. Whenever possible, we will also prove optimality of some of the proposed strategies.

While devising decontamination strategies, we pay particular attention to the impact that the choices of some parameters of the model have on the efficiency of the solutions. For example, in the case of cellular automata and mobile cellular automata, where *locality* (e.g. knowledge of neighbouring nodes) plays a crucial role, we wish to understand how important the definition of neighbourhood is to the decontamination strategy. We will then ask questions such as what happens to an optimal solution when we change the neighbourhood of a node? Can we find better strategies with larger neighbourhoods?

In particular, we will distinguish between nodes with a neighbourhood which includes only their four perpendicular neighbours (Von Neumann neighbourhood) as opposed to a neighbourhood which includes eight nodes, both the perpendicular and the diagonal neighbours (Moore neighbourhood). Another parameter that might change the computational power of the corresponding model and which is relevant in mobile cellular automata and in mobile agents is *cloning*, i.e., a particular property that allows active entities to duplicate themselves. Also, in this case we will consider how we could improve our solutions assuming the possibility of cloning.

Overall, the goal is to better understand the three models through the decontamination problem and to better understand the impact on computability and efficiency of their main parameters.

1.3 Contributions

In this thesis, we design several novel strategies for decontaminating meshes and tori networks with temporal immunity in three different models: Cellular Automata, Mobile Cellular Automata, and Mobile Agents. One of the contributions of the thesis is indeed the introduction of the mobile cellular automata model which, to our knowledge, has not been defined before. In such an environment we initiate the study on decontamination. In the other settings, the general decontamination problem has been already investigated in meshes and tori, but we are the first to consider it when nodes are subject to temporal immunity. We briefly describe the contributions in each of the three settings.

Cellular Automata Model. We describe disinfection rules in two dimensional $m \times n$ lattice ($n \leq m$) cellular automata (finite and circular) with Von Neumann and Moore neighbourhoods. The cells of the cellular automata can assume three states: *contaminated*, *decontaminating*, and *decontaminated*. Each cell changes its state at discrete time steps by applying a local rule to its own state and the states of its neighbours. Our goal is to design local decontamination rules that minimize, at each point in time, the number of simultaneously *decontaminating* cells for a given immunity time or, equivalently, the required immunity time for a given number of simultaneously *decontaminating* cells. In the case of finite cellular automata with the Von Neumann neighbourhood, we show how to achieve optimal decontamination with one, two, and four simultaneously decontaminating cells. In the case of the Moore neighbourhood, we design a set of rules for any number $k \geq 1$ of simultaneously decontaminating cells. We show that the corresponding

minimum immunity time is equal to the maximum distance between two consecutive decontaminating cells or between a decontaminating cell and a corner at time $t = 0$.

When considering circular cellular automata, we show that optimal decontamination with less than $2n$ simultaneously decontaminating cells is not possible with the Von Neumann neighbourhood and we provide a set of rules in the case of the Moore neighbourhood that doubles the number of simultaneously decontaminating cells employed in the finite case.

Mobile Cellular Automata Model. We introduce a new model, mobile cellular automata. The environment is still a two dimensional $m \times n$ lattice ($n \leq m$) that evolves in discrete time steps like a cellular automata. However, some cells are in a special active state which indicates the presence of an agent. An agent can move from node to neighbouring node but does so on the basis of the state of its neighbours. The efficiency of a decontamination strategy is measured in terms of the number of agents simultaneously present in the network. In the context of mobile cellular automata we introduce the notion of *cloning*, which allows an agent to duplicate itself and to simultaneously move in several directions. Let t_1 be the largest odd integer smaller or equal to t such that $t_1 + 1$ divides n. Let t_2 be the largest even integer smaller or equal to t such that $t_2 + 2$ divides n. Without adding the cloning capacity to the agents, we show that in the case of finite mobile cellular automata with the Von Neumann neighbourhood, when the immunity time is t, we can achieve decontamination employing $k = \min\{\frac{2n}{t_1+1}, \frac{3n}{t_2+1}\}$ agents. When considering the Moore neighbourhood, we describe a general strategy that achieves decontamination employing k agents where:

$$k = \begin{cases} \frac{2n}{t+1} & \text{if} \quad 2n \mod t+1 = 0 \\ \lfloor \frac{2n}{t+1} \rfloor + 2 & \text{Otherwise} \end{cases}$$

In the case of circular mobile cellular automata with the Von Neumann neighbourhood, when the immunity time is t, we can achieve decontamination employing $k = \min\{2\frac{2n}{t_1+1}, 2\frac{3n}{t_2+1}\}$ agents, while for the Moore neighbourhood our general strategy that achieves decontamination employing:

$$k = \begin{cases} 2 \times \frac{2n}{t+1} & \text{if} \quad 2n \mod t+1 = 0 \\ 2 \times \lfloor \frac{2n}{t+1} \rfloor + 3 & \text{Otherwise} \end{cases}$$

If the agents have cloning capabilities, on the other hand, we can achieve decontamination employing $k = \lceil \frac{2n}{t} \rceil$ agents for immunity time t in the case of finite mobile cellular

automata, and $k = 2\lceil \frac{2n}{t} \rceil$ agents in the case of circular mobile cellular automata.

Mobile Agents Model. We finally consider the network decontamination problem with temporal immunity in the classical mobile agent setting, both in meshes and tori of size $m \times n$ ($n \leq m$). While this model has been extensively investigated for the decontamination problem, it has never been studied in mesh and tori when nodes have temporal immunity. So, our work is the first in this direction. We first show that with a single agent we achieve optimal decontamination when the immunity time $t = 2m-1$ in a mesh. We then show two solutions for decontaminating a mesh with temporal immunity $t \geq 1$ that employ $k = \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$ and $k = \lceil \frac{2m-1}{t} \rceil$ agents respectively, and two solutions for decontaminating a torus with temporal immunity $t \geq 1$ that employ $k = 2 \times \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$ and $k = 2 \times \lceil \frac{2m-1}{t} \rceil$ agents respectively.

Some of the results on temporal decontamination in the Cellular Automata and Mobile Cellular Automata models have resulted in the following publications:

- Y. Daadaa, P. Flocchini, N. Zaguia. Network Decontamination with Temporal Immunity by Cellular Automata. In *9th International Conference on Cellular Automata for Research and Industry* (ACRI), pages 287-299, 2010.
- Y. Daadaa, P. Flocchini, N. Zaguia. Decontamination with Temporal Immunity by Mobile Cellular Automata. In *3rd annual Cellular Automata Workshop* (CSC), pages 172-178, 2011.

1.4 Thesis Organization

The thesis is composed of seven chapters, including the introduction provided in this chapter. The thesis is organized as follows. Chapter 2 surveys the best known problems and results related to our work in this thesis; several techniques and models employed by researchers in distributed computing and discrete dynamical systems fields are quite useful in our study. Chapter 3 presents and discusses the three different models used in this thesis: Mobile Agents, Cellular Automata, and Mobile Cellular Automata. The decontamination problem is defined according to each model specifications. Chapter 4 is devoted to temporal decontamination by cellular automata, where we consider two

dimensional finite and circular lattices with the Von Neumann and the Moore neighbourhoods. In Chapter 5, we consider temporal decontamination by mobile cellular automata. In this case the system consists of a two-dimensional lattice that evolves like a cellular automata, where however some cells are in a special active state. The active state indicates the presence of an agent, which follows a local transition rule to move from cell to neighbouring cell. Also in this setting we study decontamination with the Von Neumann and the Moore neighbourhoods, but we also introduce cloning and observe its impact on the possible solutions. In Chapter 6, we look at temporal decontamination by the classical mobile agents model in mesh and tori networks. Finally, in Chapter 7 we draw some conclusions and indicate future directions.

Chapter 2

Literature Review

The network decontamination problem is intimately related, under certain conditions, to that of the graph search. Several of the techniques and models considered by researchers on the general graph search problems are quite useful in our study of the network decontamination problems. Discrete dynamical systems, and in particular cellular automata, have been often employed to describe, model, analyze and investigate situations arising in a variety of application domains. This has recently been the case in the context of fault-tolerance of distributed systems and networks. The research in the distributed computing community on systems without decontamination or with internal decontamination, has been focused almost exclusively on the patterns of initial faults that lead to a monochromatic black fixed point (i.e., lead the entire system to a faulty behaviour). The focus is on the identification and characterization of these patterns, called dynamos, and on questions about their size. Research on the relationship between multi-agent systems and cellular automata has been focused on translation of discrete multi-agent systems into cellular automata. In this chapter, we give a survey of the most known problems and results related to our work in this thesis.

2.1 Computations by Mobile Agents in Safe Environments

In this section, we give a survey of the most known problem related to our work, where the network is assumed to be a safe environment; that is, the agents are assumed to be able to freely and safely move in the network to perform their tasks.

2.1.1 Exploration

The problem of *exploration* by mobile agents is one of the problems that can be seen as a basic block for many distributed mobile protocols. Exploration consists of having the agents collaboratively traverse an *unknown* network. The problem has been extensively studied over the past fifty years due to its various applications in different areas such as engineering, computer science, and applied mathematics. For example, tasks in environments that are not suitable for human operation, navigating a robot through a terrain containing obstacles, and finding a path through a maze. In recent years, application such as searching for data stored at unknown nodes in a computer network using mobile software agents, and obtaining maps of existing networks (e.g., computer networks, sewage systems, unexplored caves) have been studied. *Map construction* is the related problem of exploring the network to return an exact map of its topology.

Let us first consider the case of *labeled graphs*, where nodes are labeled with distinct identifiers and edges incident to a node are also labeled with distinct identifiers. Note that in this case, exploration also achieves map construction by having the agents traverse all edges of the network. Previous work on exploration of labelled graphs has emphasized minimizing the cost of exploration in terms of the total number of edge traversals (moves), and the amount of memory used by the agent [3, 30, 29, 4, 98]. In [4], Awerbuch et al. studied how a mobile robot can learn an unknown environment in a piecemeal manner. The robot's goal is to learn a complete map of its environment, while satisfying the constraint that it must return every so often to its starting position (e.g., for refuelling).

Consider now the case of *anonymous* graphs, where nodes are not labeled but edges incident to a node are labeled with distinct identifiers. An anonymous graph with locally labeled ports is a connected graph whose nodes are unlabeled, and edges incident to a node v have distinct labels $1, ..., d$, where d is the degree of v. Thus every undirected edge $\{u, v\}$ has two labels which are its port numbers at u and at v. Port numbering is local; there is no relation between labels at u and at v.

Exploration of anonymous graphs is impossible if marking of nodes is not allowed in some way. An exception is when the graph is acyclic, meaning the graph is a tree [35, 59]. Different models for marking the nodes have been used to solve the exploration problem. Pebbles which can be dropped and removed from a node was proposed by Bender et al. in [10], where it was shown that one pebble is enough to explore the graph if the robot knows an upper bound on the size of the graph, and $(\log \log n)$ pebbles are necessary and sufficient otherwise. In [45] a set of distinct markers were used to explore unlabelled

undirected graphs. In [11], Bender and Slonim used another approach consisting of employing two cooperating agents, one of which would stand on a node thus marking it, while the other explores new edges. The whiteboard model has been used by Fraigniaud and Ilcinkas [60] for exploring directed graphs focussing on minimizing the amount of memory used by the agents for exploration, and by Fraigniaud et al. [59] for exploring trees. In [59] authors also show that at least $\log n$ bits of memory are necessary for traversing a graph of size n. The agent is often modelled as a finite automaton and the emphasis is on minimizing the number of states of the automaton. In [61], Fraigniaud et al. propose a model where the robot has no *a* priori knowledge of the size or topology of the graph, and it has to explore a graph whose nodes are unlabeled and whose edge ports are locally labeled at each node. Fraigniaud et al. [61] proved that $(D \log d)$ memory bits are necessary and sufficient to explore all graphs of diameter D and maximum degree d.

Most of the known results on exploration are for a *single* agent. There have been few results on exploration by *more than one* agent [11, 59, 6, 29]. In [11], Bender and Slonim employ two agents, while in [59], Fraigniaud et al. use k agents to explore a tree. In [6], Barriere et al. assume the network is anonymous but the links are labeled with a sense of direction and the edges are locally labeled. Every edge receives two labels, one for each extremity, and all edges incident to a node receive pairwise distinct labels at this node. Authors also assume the agents are identical, their operations and movements are asynchronous, they have computing capabilities, and they communicate through whiteboards accessed in mutual exclusion. Under these conditions, Barriere et al. [6] propose a solution where a team of dispersed agents explores a graph and constructs a map, even tough the size of the network as well as the number of agents are are not known *a* priori. The proposed protocol works if the size n of the network and the number of agents k are co-prime (they have no common positive factor other than 1) and has a move complexity of $O(km)$. When the requirement of sense of direction is dropped and the agents know either n or k, the same result with the same complexity is obtained in [29].

In all works reviewed so far, the agents have their own local memory that can be used to store information on their past activities. The case of *oblivious* agents is very different, where after each movement, they forget all information about the past. In [49, 50], the exploration problem of anonymous graphs by oblivious agents has been studied in rings and trees. In [49], Flocchini et al. consider the problem of exploring an anonymous ring of size n by k oblivious anonymous asynchronous robots scattered in the ring. The robots are endowed with vision but they are unable to communicate. Within a finite

time and regardless of the initial placement of the robots, each node must be visited by at least one robot, and the robots must be in a configuration in which they all remain idle. Flocchini et al. [49] show that this problem is generally unsolvable if $k|n$ (i.e., there are initial configurations for which the exploration cannot be done). Flocchini et al. [49] then prove that, whenever $gcd(n, k) = 1$, for $k \geq 17$, the robots can explore the ring terminating within a finite time. Flocchini et al. [49] also propose a terminating protocol that explores the ring starting from an arbitrary initial configuration, and prove its correctness. Authors also consider the minimum number $\rho(n)$ of robots that can explore a ring of size n. As a consequence of the obtained positive result, Flocchini et al. [49] show that $\rho(n)$ is $O(\log n)$. Furthermore, Flocchini et al. [49] prove that $\rho(n) = (\log n)$ for infinite number of n.

In [50], Flocchini et al. studied exploration of trees by a team of robots with contrasting strengths and weaknesses, unlimited visibility on one side and obliviousness and asynchrony on the other. Essentially, Flocchini et al. [50] proved that in the case of trees, there are no general exploration protocols which are efficient. Flocchini et al. [50] showed if the tree has a maximum degree of at least 4 then you may need $\Omega(n)$ robots in order to explore it. When the trees has a maximum degree of at most 3, then you need at most $O(\frac{\log n}{\log \log n})$, and this bound is tight. Flocchini et al. [50] proved that the main reason for the complexity of the exploration problem for trees is due to the symmetries of the tree. For instance, if there are no non trivial automorphisms of the tree, 4 robots are always enough. The assumption Flocchini et al. [50] made of unlimited visibility enabled them to overcome the other computational weaknesses of the robots in the design of their exploration algorithms, in particular, the simultaneous presence of obliviousness and asynchrony.

2.1.2 Rendezvous

The rendezvous problem was posed in the 1970s. The rendezvous problem, in its basic form, asks how two unit speed players can find each other in the least expected time when randomly placed in a known dark region. The mobile agents rendezvous problem consists of $k \geq 2$ mobile agents widely dispersed on an n nodes network and trying to rendezvous in the same place within the network in a minimum amount of time. The settings differ mainly in the types and properties of the agents and the types and properties of the network. There are instances where the rendezvous problem is easy to solve, for example, if the nodes or mobile agents of the network have distinct identifiers. If the

nodes have distinct identifiers, the agents can move to a node with a specific identity, for example, the minimum. However, even in this case the problem becomes more difficult to solve if the agents do not have enough memory to remember and distinguish identities. If instead the agents have distinct identifiers, the problem is usually solved by having the mobile agents use different deterministic algorithms, thus allowing each agent to execute a different algorithm. When both the mobile agents and the network nodes are anonymous, the symmetry can make the problem difficult to solve. Many existing deterministic solutions are for agents having distinct labels but having no ability to mark the nodes of the graph.

The idea of performing a rendezvous search using marks on the starting places was introduced by Baston and Gal in [8]. If leaving marks at nodes is permitted, then it could be possible to achieve rendezvous in the anonymous setting. Two different types of marking have been proposed in the literature: whiteboards and tokens.

The rendezvous problem has been studied in both synchronous and asynchronous environments. In *synchronous* settings there is a common notion of global time and it takes exactly one unit of time for an agent to traverse an edge. Tokens were used to achieve rendezvous by two or more *anonymous agents* in *anonymous rings* by Flocchini et al. [53], Gasieniec et al. [68], and Kranakis et al. [86]. In [53], Flocchini et al. propose a solution to solve the rendezvous problem for two or more mobile agents in a ring using identical stationary tokens to break symmetry. A lower bound on the memory required for mobile agents is given. It has been proven that the mobile agent rendezvous search problem is unsolvable when the size n and the number of agents k are unknown, and it is proven that rendezvous is possible if and only if the sequence of inter-token distances is aperiodic. A detailed review of the various solutions for rendezvous in the ring is given in [84].

In [83], Kowalski and Malinowski consider the problem of deterministic synchronous rendezvous with arbitrary startup in *anonymous networks*. For rings authors presented an optimal protocol, reaching the lower bound $\Theta(n \log l)$ [32] where n is the size of the network and l is the minimum label of participating agents. For arbitrary connected graphs Kowalski and Malinowski [83] showed a deterministic rendezvous protocol polynomial in n and $\log l$, which was independent of the maximum difference of startup times. In [68], Gasieniec et al. studied the rendezvous problem with $k \geq 2$ mobile agents in anonymous rings. A new algorithm which solves the rendezvous problem for the non-symmetric distribution of agents on the ring was presented. The mobile agents require the use of $O(\log k)$ of internal memory and one indistinguishable token each. It

is assumed that all agents start to traverse the ring at the same time and they execute the same deterministic algorithm. Agents may communicate and exchange information with one another only when they meet each other at a node. All agents start to traverse the ring at the same time and they execute the same deterministic algorithm. In the periodic (but not symmetric) case, the new procedure allows the agents to conclude that rendezvous is not feasible. Note that both in the periodic as well as in the symmetric case the rendezvous cannot be completed by any deterministic procedure due to problems with breaking the symmetry.

In [63] by Pierre Fraigniaud and Andrzej Pelc, the focus is on rendezvous in trees and studying the size of memory of mobile agents which permits solving the rendezvous problem deterministically. First, authors show that if the delay between the starting times of the agents is arbitrary, then the lower bound on memory required for rendezvous is $\Omega(\log n)$ bits, even for the line of length n. This lower bound meets a previously known upper bound of $O(\log n)$ bits for rendezvous in arbitrary trees of a maximum size of n. The main result is a proof that the amount of memory needed for rendezvous with a simultaneous start depends essentially on the number l of leaves of the tree, and is exponentially less impacted by the number n of nodes. In [23] Czyzowicz et al. show the minimum amount of memory of anonymous agents that guarantees deterministic rendezvous (when it is feasible) is $\Theta(\log n)$, where n is the size of the graph, regardless of the delay between the starting times of the agents. More precisely, authors construct identical agents equipped with $\Theta(\log n)$ memory bits that solve the rendezvous problem in all graphs with a maximum of n nodes, if they start with any delay. Moreover, Czyzowicz et al.[23] prove a matching lower bound $\Omega(\log n)$ on the number of memory bits needed to accomplish rendezvous, even with a simultaneous start. In fact, this lower bound is already achieved on the class of rings.

Consider now the case of *asynchronous* networks. In [95], De Marco et al. considered the rendezvous problem in asynchronous networks, which allow a meeting of two agents passing the same edge but in the opposite directions. The proposed solution uses the knowledge of the upper bound on the size of the graph. authors first look at the case of rendezvous in an infinite line in two different scenario: when the distance between the two agents is known, and when it is unknown. The results for the infinite line carry over to the case when agents are situated in a ring of unknown size n. Authors also present a rendezvous algorithm with a cost of optimal order of magnitude working on an arbitrary unoriented ring of known size and unknown size. If the graph is arbitrary, and even when there is an upper bound on the size, the algorithm is not efficient. In [6],

Barriére et al. propose a solution using whiteboard to achieve rendezvous of multiple agents on arbitrary unlabelled graphs with a sense of direction and where each agent has a stationary token placed at the initial position of the agent. De Marco et al. [95] showed that this problem is equivalent to that of leader election. In [38]] rendezvous on an asynchronous ring with whiteboards in the presence of black holes is considered.

In [68], Gasieniec et al. studied the rendezvous problem with $k \geq 2$ mobile agents in anonymous ring. The links in the ring are unidirectional and there is a sense of direction, the size of the ring is not known to the nodes of the ring or to the agents. The k participating agents use indistinguishable tokens only once to mark their original positions in the ring, and the agents move along the ring in synchronous steps. Authors assume that all agents start to traverse the ring at the same time and execute the same deterministic algorithm. The agents may communicate and exchange information with one another only when they meet each other at a node. When an agent finds a token that has been released at a node in the ring, it cannot distinguish it from its own token(s) or from the token(s) of the other mobile agents (this is equivalent to identifying tokens with erasable marks).

In [85], Kranakis et al. studied tradeoffs between the number of tokens, memory and knowledge the agents need in order to meet in a $m \times n$ torus. Authors consider the rendezvous problem for anonymous mobile agents with tokens in a synchronous torus with a sense of direction. Kranakis et al. [85] showed that two agents with a constant number of unmovable tokens, or with one movable token each, cannot rendezvous if they have less than $O(\log n)$ memory. However agents can perform rendezvous with detection as long as they have one unmovable token and $O(\log n)$ memory. When two agents have two movable tokens each then rendezvous is possible with constant memory in an arbitrary $n \times m$ torus. Rendezvous with detection is possible with constant memory in an arbitrary $n \times n$ torus, when two agents have two movable tokens each. Finally, two agents with three movable tokens each and constant memory can perform rendezvous with detection in a $n \times m$ torus.

Rendezvous is also considered under the weak scenario where the agents are oblivious in [79, 78]. In [79], Klasing et al. consider the problem of gathering identical, memoryless, mobile robots in one node of an anonymous unoriented ring. Robots start from different nodes of the ring. Robots operate in Look-Compute-Move cycles and have to end up in the same node. In one cycle, a robot takes a snapshot of the current configuration (Look), makes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case makes an instantaneous move to this neighbour (Move). Cycles are

performed asynchronously for each robot. For an odd number of robots, Klasing et al. [79] prove that gathering is feasible if and only if the initial configuration is not periodic, and provide a gathering algorithm for any such configuration. For an even number of robots authors determine the feasibility of gathering except for one type of symmetric initial configurations, and provide gathering algorithms for initial configurations proved to be gatherable. In [78], Klasing et al. studied the gathering problem in the discrete model under the same configuration, solving it on a ring for any number of robots larger than 18. The applied technique relies on preserving symmetries. As a matter of fact, the proposed algorithm occasionally creates symmetric configurations from asymmetrical initial configurations.

2.2 Computations by Mobile Agents in Unsafe Environments

In exploration and rendezvous the network is assumed to be safe; that is, the agents are assumed to be able to freely and safely move in the network to perform their tasks. An interesting issue to consider is when either the nodes (hosts) or the agents can pose some danger to the network. In the first case, the dangerous node is a *black hole*, a node where incoming agents are trapped, and the problem is for the agents to locate it. In the second case, the dangerous agent is a *virus*, an agent moving from node to node infecting them, and the problem is for the "good" agents to capture it, that is, to *decontaminate* the network.

2.2.1 Black Hole Search

The problem of finding malicious hosts of a particularly harmful nature is known as the black holes search problem. A black hole is a node in a network which contains a stationary process destroying all mobile agents visiting this node without leaving any trace. Since agents cannot prevent being annihilated once they visit a black hole, the only protection against such processes is identifying and avoiding the hostile nodes. The only way to locate a black hole is a visit by at least one agent. An agent which falls into a black hole will be destroyed and will not turn up at a node where the other agents may expect it. This allows the surviving agents to infer the existence and location of the black hole. However, no hint about the presence of a black hole can be deduced by visiting its neighbourhood. It is also assumed that an agent visiting a black hole has

no way of communicating with other agents before being terminated. It is possible to locate a black hole only by sacrificing one agent and by using another agent to indirectly infer the existence of a black hole. An agent which is to visit an unknown node can, for instance, have a meeting scheduled with another agent after the visit, or write on a whiteboard in a neighbouring node the label of the unknown node that he is visiting. If the visited node is a black hole, then the destroyed agent will neither turn up at the node where the meeting was scheduled nor write back to the whiteboard that the node has been successfully visited. In both cases, the surviving agents can deduce that the visited node is a black hole. In this scenario there is an upper bound on the time needed by an agent for traversing any edge safely.

In the last decade, there has been a great deal of work done on finding faults in networks using mobile entities. Most of the existing work deasl with the black hole search problem for a single black hole [24, 25, 36, 37, 38, 39, 40, 41, 42, 43, 44, 51, 70, 80, 81]. Some work has been done for multiple black holes in [22, 82], although both of these papers use the synchronous model, as do some of the single black hole papers [24, 25, 80, 81]. Existing solutions can be classified based on the network model: synchronous and asynchronous.

The *asynchronous* scenario was studied under different agent models (i.e., network knowledge, tokens, whiteboard). In [40], Dobrev et al. provided solutions to the black hole search problem in anonymous rings using whiteboards for two settings; when the anonymous agents are co-located, and when they are dispersed. They proved that two such agents are necessary and sufficient to locate the black hole. In [39], Dobrev et al. provide a characterization of the impact that factors such as a priori network knowledge and consistency of the local port labelling have on the complexity of the black hole location problem. Dobrev et al. [39] consider both the case of topological ignorance in systems where there is sense of direction and the case of complete topological knowledge of the network. Authors show that, in both cases, two agents suffice.

In [36, 41, 42], Dobrev et al. investigate the black hole search problem for two agents with a map and whiteboard searching for a single black hole. In [41], Dobrev et al. present a search protocol that improves the bounds from the worst case lower bound of $\Omega(n \log n)$ agent moves to $O(n + d \log d)$ agent moves, where d is the diameter of the network. The result allows for $\Theta(n)$ moves for a large class of possibly unstructured networks with low diameter. In [36], Dobrev et al. show that with a map of the network, a team of two agents suffice, and the number of moves is in the worst case $O(n \log n)$. They also present a general strategy that allows two agents to locate the black hole with $O(n)$

moves in common interconnection networks such as hypercubes, cube-connected cycles, star graphs, wrapped butterflies, and chordal rings, as well as in multidimensional meshes and tori of restricted diameter. These results hold even if the networks are anonymous. In [42], Dobrev et al. use a technique based on pre-calculating the open vertex cover of cycles of a graph that allows them to solve the problem in $\Theta(n)$ for a large class of networks.

In contrast, in [70], Glaus studied the black hole search problem without the knowledge of the incoming link, and has shown that this modification has effects on the size of the solution. Glaus [70] provided the lower bound on the number of agents that are necessary to locate the black hole; any correct algorithm solving the black hole search problem without the knowledge of the incoming link needs at least $\frac{\Delta^2+\Delta}{2}+1$ agents. The algorithm uses the optimal number of agents in the worst case, however, the cost of the algorithm and bounds on the optimal cost of the solution were not shown in this paper.

In [43], Dobrev et al. consider the token model, where each agent has a bounded number of tokens available that can be carried, placed on, or removed from a node. All tokens are identical (i.e., indistinguishable), and no other form of communication or coordination is available to the agents. Authors first prove that a team of two agents is sufficient to locate the black hole in finite time even in this weaker coordination model. Furthermore, Dobrev et al. [43] prove that this can be accomplished using only $O(n \log n)$ moves in total, which is optimal, the same as with whiteboards. Finally, authors show that to achieve this result the agents need to use only $O(1)$ tokens each. The previous strategy is generalized in [37], where Dobrev et al. look to the case of unknown graph. Authors present an algorithm that works in the token model and solves the black hole search problem with the minimal number of agents and with a polynomial number of moves. Dobrev et al. [37] algorithm works even if the agents are asynchronous, and if both the agents and the nodes are anonymous. More precisely, authors consider an unknown, arbitrary, anonymous network and a team of exploring agents starting their identical algorithm from the same node (homebase). The agents are anonymous, and they move from node to neighbouring node asynchronously. Each agent has an indistinguishable token (or pebble) available that can be placed on, or removed from a node. The token can be placed on a node, either in the center or on an incident link. In the proposed algorithm, two tokens are never placed in the same location (node center or port), nor does an agent ever carry more than one token. Using only this tool for marking nodes and communicating information, authors show that with $\Delta + 1$ agents (where Δ is the maximal degree of the graph), the exploration can be successfully completed. The

proposed algorithm allows at least one agent to survive and, within a finite time, the surviving agents will know the location of the black hole with the allowed level of accuracy. The number of moves performed by the agents when executing the proposed protocol is shown to be polynomial, and the proposed algorithm is rather complex.

In [44], Dobrev et al. show not only that a black hole can be located in a ring using tokens with scattered agents, but also that the problem is solvable even if the ring is un-oriented. First authors prove that the black hole search problem can be solved using only three scattered agents. Then, Dobrev et al. [44] show that, with $k(k > 4)$ scattered agents, the black hole can be located in $O(kn + n \log n)$ moves. Moreover, when $k(k > 4)$ is a constant number, the move cost can be reduced to $O(nlogn)$, which is optimal. These results hold even if both agents and nodes are anonymous. In [38], Dobrev et al. consider k anonymous, asynchronous mobile agents in an anonymous ring with a black hole. The agents are aware of the existence, but not of the location of such a danger, and the network is totally asynchronous. In this setting it was observed that in order to solve the problem, the network must be 2-connected. A black hole search is not feasible in trees, because in asynchronous networks it is impossible to distinguish a black hole from an incident slow link. The only way to locate a black hole is to visit all other nodes and learn that they are safe. In particular, it is impossible to answer the question of whether a black hole actually exists in the network, hence authors worked under the assumption that there is exactly one black hole and the task was to locate it. In [51], Flocchini et al. prove that the pure token model is computationally as powerful as the whiteboard model for the black hole search problem. Furthermore, the complexity is exactly the same. Authors prove that a team of two asynchronous agents, each endowed with a single identical pebble (which can be placed only on nodes, and with no more than one pebble per node) can locate the black hole in an arbitrary network of known topology. This can be done with $(n \log n)$ moves, where n is the number of nodes.

In [80], Klasing et al. consider the problem of designing a black hole scheme under the scenario of synchronous networks. Authors investigate the case when there may be at most one black hole in the network. The search is performed by exactly two agents, which start from the same node homebase and can communicate only when they are in the same node. At least one agent must report the information back to the homebase on the exact location of the black hole or whether one exists. In [24], Czyzowicz et al. studied the black hole problem in a (partially) synchronous network, assuming an upper bound on the time of any edge traversal by an agent. The minimum number of agents capable to identify a black hole is two for a given graph and a given starting node.

Czyzowicz et al. [24] looked to the fastest possible black hole search by two agents, under the general scenario in which some subsets of nodes are safe and the black hole can be located in one of the remaining nodes. Authors show that the problem of finding the fastest possible black hole search scheme by two agents is NP-hard, and they give a polynomial approximation to solve it.

In [25], Czyzowicz looked at the same problem in trees, and gave optimal black hole search algorithms for two extreme classes of trees: the class of lines and the class of trees in which any internal node (including the root which is the starting node) has at least two children. In [81], Klasing et al. consider the same problem assuming that the map of the network is given. Their objective is minimizing the number of agents that fall into the black hole and the time taken by the surviving agents to locate the black hole. The proposed algorithm explores the network via a spanning tree. In [22], Cooper et al. were the first to consider the general case of multiple black holes using k agents starting from the same node. The agents move through the network in synchronous steps and can communicate only when they meet in a node. In [82], Kosowski et al. look at a multiple black hole search: assuming a directed graph. The robots are associated with unique identifiers, they know the number of nodes in the graph (or at least an upper bound), and they know the number of edges leading to the black holes. Each node is associated with a whiteboard, separately considering the synchronous and the asynchronous cases.

2.2.2 External Decontamination

In this part we review results related to the decontamination problem. We first briefly introduce an earlier related problem, the graph search problem. We then focus on the problem considered in this thesis: deploying a team of agents to decontaminate a network possibly contaminated by a virus.

Graph Search. The decontamination problem considered in this thesis is a variation of a problem extensively studied in the literature known as *graph search*. The graph search problem was first introduced by Breish in [16], where an approach for the problem of finding an explorer that is lost in a complicated system of dark caves is given. In [99, 100], Parsons proposed and studied the pursuit-evasion problem on graphs. Members of a team of searchers traverse the edges of a graph in pursuit of a fugitive, who moves along the edges of the graph with complete knowledge of the locations of the pursuers. Notice that in the classical graph search problem the agents can arbitrarily jump from

a node to any other node in the graph. The problem of graph search is formulated as follows. Given a contaminated connected network, via a sequence of operations by using searchers, the purpose is to achieve a state in which all nodes and edges are simultaneously decontaminated.

There are two versions of the graph search problem namely, node search and edge search depending on whether the nodes or the edges are contaminated in the graph. In the node searching problem, possible operations are placing a searcher on a node and removing a searcher from a node. In addition to the operations possible in node searching problem, one more operation is possible in edge searching problem, moving a searcher along an edge. A contaminated edge is cleared by moving a searcher along this edge. The efficiency of a graph search solution is based on the size of the searching team. This number is called the node search number ($ns(G)$) or the edge search number ($es(G)$) respectively in the node or edge search problem. In [96], Megiddo et al. proved that for arbitrary graphs, determining if the search number is less than or equal to an integer K is an NP-complete problem. In [91], LaPaugh proved that monotone searching was possible for every graph, that is, it is possible to design a solution for every graph where once a node is cleaned it does not have to be cleaned again.

Decontamination. The main difference in the setting described in this thesis is that the agents, which are pieces of software, cannot be removed from the network; they can only move from a node to a neighbouring node. This additional constraint introduced and first studied by Barrière et al. in [5] resulting in a contiguous, monotone, node search or intruder capture problem The contiguous monotone decontamination strategy is defined as follows: (1) the agents cannot be removed from the network; (2) at any time of the search strategy, the set of clean nodes forms a connected subnetwork; and (3) a clean node cannot be recontaminated. The continuous assumption is motivated by the fact that the software agents that are only able to move on the edges of the network changed the nature of the problem of graph search, and therefor the classical results on node and edge search do not generally apply. In [7], Barrière et al. prove that the search number in a continuous model is equal to or greater than the search number of the original non-continuous model, and this problem is NP-hard. In the rest of this section we use the term decontamination to refer to contiguous monotone node search as defined in [5].

The model for decontamination studied in the literature is defined as follows. A team of agents is initially located at the same node, the homebase, and all the other nodes are

contaminated. A decontamination strategy consists of a sequence of movements of the agents along the edges of the network. The agents can communicate when they reside on the same node. At any point in time each node of the network can be in one of three possible states: clean, contaminated, or guarded. A node is guarded when it contains at least one agent. A node is clean when an agent passes by it and all its neighbouring nodes are clean or guarded, contaminated otherwise. Initially all nodes are contaminated except for the homebase (which is guarded). The solution to the problem is given by devising a strategy for the agents to move in the network in such a way that at the end all the nodes are clean.

Starting from the classical model introduced in [5] (called the *Local Model*), additional assumptions have sometimes been added to study the impact that more powerful agents or systems capabilities have on the solutions to the decontamination problem. One assumption is that an agent located at a node can see only local information, like the state of the node, the labels of the incident links, and the other agents present at the node. Another assumption is visibility which is the capability of the agent to see the state of its neighbours, i.e., an agent can see whether a neighbouring node is guarded, clean, or contaminated. In some mobile agent systems, the visibility power could be easily achieved by probing the state of neighbouring nodes before making a decision. A third assumptions is cloning which is the capability of an agent to clone copies of itself. Lastly, synchronicity, which implies that local computations are instantaneous, and it takes one unit of time (one step) for an agent to move from a node to a neighbouring one. There is a large body of work on decontamination in the mobile agent model [92, 5, 73, 47, 57, 46, 48, 47, 14, 94, 55, 56, 58]. However, all but two, [14, 58], focus on specific topologies such as trees, hypercubes, meshes, and tori.

The Tree. The tree was the first topology to be investigated in the *Basic Model*. In [5], Barrière et al. showed that for a given tree T, the minimum number of agents needed to decontaminate T depends on the location of the homebase. The proposed solution is based on two observations. Consider node A, if A is not the homebase, the agents will arrive at A for the first time from some link e. Let $T_1(A), \ldots, T_i(A), \ldots, T_{d(A)-1}$ be the subtrees of A from the other incident links, where $d(A)$ denotes the degree of A, let m_i be the number of agents needed to decontaminate $T_i(A)$ once the agents are at A, and let $m_i \geq m_{i+1}$, $1 \leq i \leq d(A)-2$. The first observation is that to decontaminate A and all its other subtrees without recontamination, the minimum number $m(A)$ of agents needed is $m(A) = m_1$ if $m_1 > m_2$ and $m(A) = m_1 + 1$ if $m_1 = m_2$. Consider now homebase B, let

$m_j(B)$ be the minimum number of agents needed to decontaminate the subtree $T_j(B)$, and let $m_j \geq m_{j+1}, 1 \leq j \leq d(B)$. The second observation is that to decontaminate the entire tree starting from B the minimum number $m(B)$ of agents needed is $m(B) = m_1$ if $m_1 > m_2$ and $m(B) = m_1 + 1$ if $m_1 = m_2$.

Based on these two observations, Barrière et al. [5] first show how the determination of the optimal number of agents can be done through a saturation. Simple informations about the structure of the tree are collected from the leaves and propagated along the tree, until the optimal number of agents is known for each possible starting point. The most interesting aspect of this strategy is that it immediately yields a protocol for trees that uses the exact minimum number of agents. The technique to determine the minimum number of agents and the corresponding decontamination strategy is done in $O(n)$ time and exchanges $O(n)$ messages. The algorithm is also naturally distributed, the minimum number of agents and the decontamination strategy can be computed in a decentralized manner. The trees that require the largest number of agents are complete binary trees, where the number of agents is $O(\log n)$. In contrast, in the line two agents are sufficient.

The Hypercube. Decontamination in a hypercube has been studied in [47], in which Flocchini et al. prove a lower bound on the number of agents necessary and sufficient to decontaminate a hypercube of size n, $\Theta\frac{n}{\sqrt{\log n}}$. The employ of this optimal number in the *L*ocal Model has an interesting consequence, $\Theta\frac{n}{\sqrt{\log n}}$ is the search number in the classical graph search problem. Four different strategies were proposed. In the first strategy (*Local model*), one of the agents acts as a coordinator for the entire cleaning process. The cleaning strategy is carried out on the broadcast tree of the hypercube. The main idea is to place enough agents on the homebase and to have them move, level by level, on the edges of the broadcast tree, led by the coordinator in such a way that no recontamination may occur. This strategy employs an optimal number of agents ($\Theta\frac{n}{\sqrt{\log n}}$), with $O(n \log n)$ moves, and runs in $O(n \log n)$ time steps. The second strategy is devised for a model where the agents are allowed to "see" the state of their neighbours. In this case, the computation is local, so there is no need for a coordinator and agents can move autonomously. In fact, the agents are still moving on the broadcast tree, but they do not have to follow the order imposed by the coordinator. In this setting the solution requires $\frac{n}{2}$ agents, but the time complexity is optimal ($\log n$ time steps), and requires the same number of moves ($O(n \log n)$). Finally, the last two strategies are devised for models that assume agents have cloning capabilities and either can "see" the state of their neighbours or move in a synchronous setting. In both cases the bound on

the number of moves becomes optimal decreasing to $n-1$.

The Mesh. In [55], Flocchini et al. consider the problem of decontaminating a $m \times n-$Mesh ($m \leq n$). They show some lower bounds on the number of agents, number of moves, and time required to decontaminate an $m \times n-$Mesh ($m \leq n$). At least m agents, mn moves, and $m+n-2$ time units are required to solve the decontamination problem. The authors consider two models, one in which an agent has only local knowledge about the node where it resides, and the other in which an agent has visibility to see their neighbouring nodes.

In the first model, the only knowledge that an agent has is the information written on the local whiteboard at its current location. The algorithm is described as follows. In the initialization phase, all agents have entered the network in the homebase which is the initial position ($P(0,0)$, upper left most node in the mesh), before the cleaning. Each searching agent is informed by a Synchronizer S to move to its starting point along the first column, but S stays at node $P(0,0)$. By the end of this step there will be one searcher in each node of the first column of the Mesh, while S and one of the searching agents are at the node $P(0,0)$. In the cleaning phase; the Synchronizer S moves SOUTH and forces the searching agents to move EAST one at a time. When the whole column of agents has moved to the next column, the Synchronizer will also move EAST to the next column. Then, it will move NORTH and continue to force the searching agents to move EAST one at a time. Again, when the whole column of agents have moved to the next column, the Synchronizer will also move EAST to the next column. These operations are repeated until the Mesh is cleaned. This algorithm requires $m+1$ agents, $\frac{m^2+4mn-5m-2}{2}$ moves, and $(mn-2)$ time units.

In the second model, agents have the power of visibility. Each agent can see the other agents located at its adjacent neighbouring nodes and may coordinate its searching operations according to its neighbouring agents‘ moves. In other words, each agent moves independently without the need of a Synchronizer, and agents communicate with each other by using the whiteboard associated with each node in the Mesh. In the initialization phase, all the m agents are located at the node $P(0;0)$. In the cleaning phase, all agents wake up to be the searchers s. Each searcher s will independently perform the following operations: s reads the whiteboard on the current node. If the whiteboard has a “CLEAN” message, s moves SOUTH to the next row. s will contiguously move SOUTH until it reaches a node on which the Whiteboard is empty. If the whiteboard is empty, s writes a “CLEAN” message on it. s guards the current node until it can see that

its neighbouring nodes NORTH-NORTH, NORTH-WEST, and NORTH-SOUTH except NORTH-EAST are all clean (or guarded), then s moves EAST to the next column. s will repeat this operation until it reaches the last column in the Mesh. The Mesh is cleaned when all the m searching agents reach the last column. This algorithm requires m Agents, $\frac{m^2+2mn-3m}{2}$ moves, and $(m+n-2)$ time unit, time and number of agents complexity are optimal.

Tori and Chordal Rings. In [48], Flocchini et al. studied the decontamination problem in tori and chordal rings. It has been shown that any solution of the decontamination problem in a torus $T(h,k)$ with $h,k \geq 4$ requires at least $2 \times min(h,k)$ agents, and in the *L*ocal Model it requires at least $2 \times min(h,k)+1$ agents. To match the lower bound a very simple strategy is employed. The idea is to deploy the agents to cover two consecutive columns and then keep one column of agents to guard from recontamination and have the other column move along the torus. In this setting the solution requires $2h+1$ agents, $hk-2h$ time units and $2hk-4h-1$ moves, where h,k are the dimensions of the torus, hk. As for the other topologies, visibility decreases time and slightly increases the number of agents. In the case of torus, it is interesting that in the *V*isibility Model all three complexity measures are optimal. This strategy employs $2h$ agents, $\lceil \frac{k-2}{2} \rceil$ time units and $hk-2h$ moves. These strategies were generalized to the case of d-dimensional tori.

The *L*ocal and *V*isibility Models have been also studied in the chordal ring topology. A chordal ring with n nodes is defined as $C(\langle d_1 = 1, d_2, ..., d_k \rangle)$ and a link structure is defined as $(\langle d_1 = 1, d_2, ..., d_k \rangle)$ where $d_i < d_{i+1}$ and $d_k \leq \lfloor \frac{n}{2} \rfloor$. In [48], it is first shown that the smallest number of agents needed for the decontamination does not depend on the size of the chordal ring, but solely on the length of the longest chord. In fact, any solution of the contiguous decontamination problem in a chordal ring $C(\langle d_1 = 1, d_2, ..., d_k \rangle)$ with $4 \leq d_k \leq \sqrt{n}$ requires at least $2d_k$ agents in the *L*ocal Model and $2d_k+1$ agents in the *V*isibility Model. In both models, the cleaning is preceded by a deployment stage after which the agents have to occupy $2d_k$ consecutive nodes. After the deployment, the decontamination stage can start. In the *L*ocal Model, nodes x_0 to $x_{d_k ? 1}$ are constantly guarded by one agent each, forming a window of d_k agents. This window of agents will shield the clean nodes from recontamination from one direction of the ring while the agents of the other window are moved by the coordinator (one at a time starting from the one occupying node x_{d_k}) along their longest chord to clean the next window in the ring. Also in the case of the chordal ring, the visibility assumption allows the agents to

make their own decision solely on the basis of their local knowledge. An agent move to clean a neighbour only when this is the only contaminated neighbour.

Temporal Immunity. In [56], Flocchini et al. introduce decontamination with *temporal immunity* in a tree, which is the model of decontamination used in this thesis. The main difference between the classical decontamination model, and the new model is that, a cleaner is able to decontaminate any infected node it visits. Once the cleaner departs, the decontaminated node is immune for a certain $t \geq 0$ (i.e. $t = 0$ corresponds to the model without temporal immunity studied in the previous work) time units to viral attacks from infected neighbours. After the immunity time t is elapsed, recontamination can occur. The minimum team size necessary to disinfect any given tree with immunity time t is derived. Further, Flocchini et al. [56] show how to compute the minimum team size for all nodes of the tree and implicitly the solution strategy starting from each starting node. These computations use a total of $\Theta(n)$ time (serially) or $\Theta(n)$ messages (distributively). Authors then provide a complete structural characterization of the class of trees that can be decontaminated with k agents and immunity time t; Flocchini et al. [56] do so by identifying the forbidden subgraphs and analyzing their properties. Finally, authors consider generic decontamination algorithms, protocols that work unchanged in a large class of trees with little knowledge of their topological structure. Flocchini et al. [56] prove that, for each immunity time $t \geq 0$, all trees of a maximum height h can be decontaminated by a team of $k = \lfloor \frac{2h}{t+2} \rfloor$ agents whose only knowledge of the tree is the bound h.

2.3 Cellular Automata and other discrete dynamical system

The concept of discrete dynamical systems known as cellular automata was first proposed by Von Neumann and Ulam in the early 1950‘s as models of self-organizing / reproducing behaviours. Since then, cellular automata have been used to model various dynamical processes in biology, chemistry, and physics. The simple structure of cellular automata has attracted researchers from various disciplines. It has been subjected to rigorous mathematical and physical analysis for the last fifty years and its application has been proposed in different branches of science both physical and social.

2.3.1 Cellular Automata

The popularity of cellular automata can be traced to their simplicity, and to the enormous potential they hold in modelling complex systems. Cellular automata consist of a series of identical components, also called cells, whose states are synchronously updated following a rule applied to a predefined neighbourhood of each cell. As originally defined, cellular automata were discrete in space, time and state. It was quickly seen that even with simple rules and binary states, cellular automata could exhibit what appeared to be quite complex behaviours and were capable of universal computation (i.e., can have the power of a Turing machine). The most famous example is the Game of Life which was proposed by John Conway in 1970. In the Game of Life binary state cellular automata where the cells are connected in a 2-dimensional grid and each cell's neighbourhood η_i^t consist of its 4 direct neighbours and the 4 diagonal neighbours in the grid (Moore neighbourhood). Let s_i^t be the state of cell i at time t. The transition rule ϕ that transforms a state s_i at time t into the next state at time $t+1$ on the basis of its neighbourhoodη_i^t is very simple. If $s_i^t = 1$, then $\phi(\eta_i^t) = 1$ if and only if exactly two or three other neighbours are in state 1, otherwise $\phi(\eta_i^t) = 0$. If $s_i^t = 0$, then $\phi(\eta_i^t) = 1$ if and only if exactly three other neighbours are in state 1; otherwise $\phi(\eta_i^t) = 0$. Even from these simple rules, amazing patterns appeared that seemed to mimic structures found in nature; gliders that floated across the screen, still life patterns that remained eternally fixed, and oscillators that ran through a sequence of patterns before returning to their original form. It is not surprising then that cellular automata have been used in a wide array of disciplines to mimic and predict complex behaviours that eluded simple descriptions using other areas of mathematics such as differential equations.

Elementary cellular automata are the simplest possible cellular automata. They are obtained by considering only a one dimensional grid (i.e., a line of cells), two possible states and nearest neighbours (i.e., left and right one in addition to the cell itself). Based on these simple parameters, there are 256 distinct possible local rules that exhibit a wide range of behaviours and properties. The richness and diversity of these simple rules has engendered an extensive amount of research since their introduction. They have been studied from many different perspectives, for example, for their dynamical behaviour [1, 2, 31, 87, 107], algebraic properties [75, 74, 76], topological properties [17, 18, 19], and from a computational complexity point of view [28, 33, 34]. One of the authors who has most deeply studied the behaviour and properties of cellular automata is Wolfram [109].

2.3.2 Ants and Turmites

In the mid-1980s, in [90] Langton defined several discrete systems modelling several aspects of living beings and, in particular, he defined *ants*. An ant is defined as an automaton moving on a square planar grid whose vertices are coloured either in black or white. When the ant enters a vertex, it decides the direction in which it will move according to the colour of the vertex. It turns to its left if the vertex is black and to its right otherwise, and after that, it flips the colour of the vertex. When the ant starts moving on a grid whose vertices are all the same colour, it engages in an apparently chaotic behaviour for about 10000 steps, and then it falls into a periodical movement with a drift. This unexpected behaviour is what Langton was interested in. The same model was also extensively studied in physics. From the physicists point of view, the ant is described by its position and its velocity. Several extensions to ants are given in the literature; extension to multiple colours, extension to multiple ants, and extension to multiple states.

A further extension of Langton's ants is to consider multiple states of the Turing machine as if the ant itself has a colour that can change. These late ants are called *turmites*, a contraction of "Turing machine termites". Common behaviours include the production of highways, chaotic growth, and spiral growth. Turmites follow simple local rules [9], but their global behaviour is generally complex, even when only one agent is used. The complexity of the system has been well studied with a single turmite, either with the original rule [67, 15, 66] or with generalizations [65, 64]. However, systems with multiple turmites have been explored much less [21, 13, 9]. One reason why the multi-turmite system has been scarcely studied is that introducing multiple turmites also produces ambiguities with regard to how to update agents and how to solve their potential conflicts. In some cases, ambiguities may even render experiments difficult to reproduce.

2.3.3 Multi Agents Systems and Cellular Automata

The problem of translating the discrete multi agent systems into cellular automata has been introduced in the last few years [20, 108]. Contrarily to the sequential scheduling generally used in mobile agent systems simulations, cellular automata are a model for massively parallel computing where the updating of the components is synchronous. However, cellular automata expressiveness is limited and not always adapted to build models where independent entities move and act on neighbour cells. After illustrating

these issues on a simple example in [108], Spicher et al. propose a generic method to translate a discrete mobile agents system into a cellular automata, called a transactional cellular automata. The advantage of using the cellular automata form is simplicity; it involves static homogeneous computing units that are regularly arranged in space. The drawback of expressing a model with cellular automata appears when one needs to build models with pseudo-independent entities that may move and act on neighbour cells. Indeed, in cellular automata, a cell cannot directly change the state of its neighbour cells, whereas such an ability is usually required to express a mobile agent system model.

In [20], Chevrier et al. propose a mathematical description of multi agent systems as discrete dynamical systems. The key idea is that agents should never act directly on other components of the system (agents or environment), but release influences which are then combined to update the state of the system. Authors propose a method which decomposes the definitions of a multi agent system into six parts: (1) the basic sets, (2) the perception of the agents, (3) the influences, (4) the updating of the agents internal state, (5) the updating of the environment, and (6) the updating of the position and observable states of the agents. Chevrier et al. [20] illustrate the proposed method on the multi turmite model, also known as the multiple Langton‘s ants model. A turmite or ant can be viewed as agents that evolve on a grid. Their actions are limited to moving forward, turning left or right and, inverting the state of the cell on which they are located (an operation that we call flipping the cell).

2.3.4 Dynamos and Internal Decontamination

With internal decontamination we refer to a dynamical process in cellular system, where a binary states are associated to the cells (e.g., contaminated (faulty) and decontaminated (non-faulty)) and local rules describe the spread of decontamination and / or contamination. In this setting a problem that has been investigated is the determination of patterns of contaminated cells (initial state configurations) such that their evaluation under majority rules do create monochromatic fixed points,. Often the goal has been to determine the one of minimum size. Such patterns are called *dynamos*.

Let $G = (V, E)$ be a simple undirected connected graph of size n where nodes $v_1, v_2, ...v_n$ are coloured black or white (i.e., have boolean states). Let $x(v_i) \in 0, 1$ be the state of node v_i, where 0 corresponds to white and 1 corresponds to black. A node subject to majority voting updates its colour, assuming the colour held by the majority of its neighbours. The update is performed simultaneously at discrete time steps by all

nodes subject to majority voting, that is, the dynamics are synchronous. An irreversible dynamo is one where the initial black vertices do not change their colour regardless of their neighbourhood, while in reversible dynamos nodes may switch colour several times according to a changing neighbourhood. A subclass of the later monotone reversible dynamos are the ones for which black nodes never become white because the neighbourhood never forces them to change colour.

When internal decontamination mechanisms are in place, a local faulty behaviour can be mended by the existing local-majority mechanism, that is, in systems with internal decontamination, all nodes are subject to the majority rule. In this case, dynamos are also called reversible. This is the classical model studied in most of the literature. The majority rule has been studied extensively in the context of discrete dynamical systems. Let $X^t = (x^t(v_1), ...x^t(v_n))$ be the global configuration at time t of the n vertices $v_1, ...v_n$ of a graph, where the majority rule is applied synchronously at discrete time steps to all vertices. It has been shown in [72] by Goles and Olivos that for finite graphs, any sequence X^t reaches a fixed point or a cycle of length two. In [104, 105, 71] this interesting behaviour has been actually shown to hold for a more general setting, for example, when the majority function is replaced by a threshold function and when the edges have weight. In [97], Moran generalized this property for majority rules over locally finite connected graphs, for which a sufficient condition is given in terms of the rate of growth of the graph. The property has been generalized further to include local periodicity [69], and also in this more general case, a sufficient condition is provided.

In [102], Peleg introduced the study of reversible dynamos in distributed computing, and the concern mainly has been the determination of the smallest size for a dynamo. His work started with the study of 1-time dynamos, where the monochromatic fixed point has to be reached in a single step. A variety of results, mostly on the minimum size of 1-time dynamos both in general graphs and in specific topologies have been derived. A more general case when the majority is performed on a $r-$neighbourhood has also been considered. In the definition of majority voting it is very important to define what action a node should take in case of tie. Possible actions are related in [101]. Two plausible options would be to give priority to one specific colour, without lose of generality white, or to give priority to the current colour of the vertex (i.e., require strict majority in order to change the current colour). Another (perhaps less well-motivated) option is to give priority to flipping the current colour in case of a tie. These three options are denoted by, prefer-white (PW), prefer-black (PB), prefer-current (PC), and prefer-flip (PF) respectivelly. Another distinction regards whether the node making the decision

is included in the computation of the majority, self-including (SI), self-not-including (SN). Depending on the combination of the various parameters, different models, often with quite different dynamics, can be defined (sometimes indicated as (PW,SI), (PB,SI), etc). The model (PB,SN) is usually called a simple majority, while (PC,SN) is called a strong majority. It is very clear that requiring monotonicity in the process also strongly influences the dynamics. For example, Peleg proved in [102] a lower bound of $\Omega(\sqrt{n})$ on the size of monotone dynamos in most variant of the models, while in [12] Berger proved that without imposing monotonicity, for every $n \geq 1$ there exists a graph G of n or more vertices with a dynamo of size $O(1)$ in all models. Some specific topologies have also been studied in the reversible model, but always with the monotonicity condition, that is when decontamination is ineffective. In this context, tori [54] and some interconnection networks have been studied, where trade-offs between time and size have been determined [52].

In [93] Luccio et al. studied the network decontamination problem under a new model of immunity to recontamination. A clean node, after the cleaning agent has gone, becomes recontaminated only if a weak majority of its neighbours are infected. This recontamination rule is called local immunization. Luccio et al. [93] study the effects of this level of immunity on the nature of the problem in tori and trees. More precisely, they establishe lower-bounds on the number of agents necessary for decontamination, and on the number of moves performed by an optimal-size team of cleaners, and propose cleaning strategies. The bounds are tight for trees and for synchronous tori; they are within a constant factor of each other in the case of asynchronous tori. It is shown that with local immunization only $O(1)$ agents are needed to decontaminate meshes and tori, regardless of their size. This must be contrasted with the $2minn, m$ agents required to decontaminate a $n \times m$ torus without local immunization [48]. Interestingly, among tree networks, binary trees are the ones requiring the highest number of agents ($O(\log n)$) in the worst case without local immunization [5]. Instead, with local immunization, they can be decontaminated by a single agent.

Chapter 3

Model

In this chapter we introduce the three models used in this thesis. We first describe the *Mobile Agents Systems* (MA), a model of mobile agents commonly employed in the distributed computing community, where powerful autonomous computational entities move and interact on a network. We then consider the *Cellular Automata* (CA) model. In this model there are no active entities moving between cells. Instead there is a cellular space where cells change their state synchronously according to local rules. Finally, we describe the *Mobile Cellular Automata* (MCA), a combination of CA and MA models. This model is typically studied in the artificial intelligence and discrete systems communities where simple entities possessing visibility of their neighbourhood move synchronously on a cellular space following local rules. In the discussion of each model, we will consider the problem of decontaminating the environment.

3.1 The Decontamination Problem

In this thesis we develop strategies for the decontamination of grids, tori, and chordal rings. In all three models the environment is considered initially contaminated (possibly by a virus) and the goal is to simultaneously decontaminate all nodes, minimizing the resources of the system. The various assumptions of each particular model leads to different decontamination rules, however, contamination is common to all three models. The system has a given *immunity time* $t > 0$ and as soon as a node is decontaminated, it will stay decontaminated regardless of the state of its neighbours for t time units. After that time has elapsed, the node will be recontaminated if at least one neighbour is contaminated. In other words, nodes are immune for a limited amount of time and

contamination spreads from node to neighbouring non-immune nodes.

A decontamination strategy is said to be *monotone* if once a node becomes decontaminated, it does not become contaminated ever again. In most instances of the decontamination problem there exist monotone optimal strategies. In this thesis we focus only on monotone strategies.

Another common feature to our three models is that both contamination and decontamination spread from node to neighbouring node. In particular, when agents are involved, they may start from separate locations (homebases) but they move from node to neighbouring node without "jumping".

As mentioned in Chapter 2, the decontamination problem has been studied in several contexts but very little is known when nodes have temporal immunity. When describing the details of each of our three models we will also explain the relationship with the related work. In the following we describe the details of each individual model.

3.2 The Mobile Agents Model

The Agents. Mobile agents are active computational entities that move on a network. As seen in Chapter 2, in the literature, they have been studied under a variety of assumptions regarding their environment and capabilities.

Agents can communicate with each other in different ways. In the *face-to-face* model, agents can communicate only when they reside on the same node, whereas in the *token* model agents exchange information by using tokens to be placed and picked up at nodes. In the *whiteboard* model, each node contains a whiteboard (a storage area) which can be accessed in fair mutual exclusion by the agents for writing and reading information. In this thesis we consider the *whiteboard* model. A distinction that is commonly made among different models is whether or not agents have distinct identifiers. In our setting the agents have distinct identifiers.

The agents have computing capabilities and private storage, although the computational power depends on the amount of storage they may have. As seen in the previous chapter, the focus of the existing work ranges from agents with no memory at all (see [50, 49, 78]), to agents with constant memory (see [23, 56, 60]), to agents with no restriction on the memory size (see [32, 59, 98]).

As in most recent algorithmic work employing mobile agents (see [5, 47, 48]), in the MA model the agents can only move from node to neighbouring node as opposed to being able to jump from a node to any other node in the network (see [77, 100, 99]).

When at a node, agents perform local computations according to a predefined set of behavioural rules. These rules are the same for all agents and depend on the content of the whiteboard and the local memory. In this thesis we do not restrict the amount of local memory available to the agents.

An agent usually has a *homebase*, that is, a node from which it originated. Whether agents all start from the same homebase (co-located agents) or from different homebases usually makes a difference in the design and efficiency of protocols. In this mobile agent model, agents start from *a single locations*.

Another factor that differentiates the various models studied in the literature is synchronicity of movement. In this thesis we assume that the agents move *synchronously* and simultaneously (as in [56, 92, 93]), meaning each agent is active at discrete time steps and it takes one unit of time for an agent to traverse a link. Several studies focus instead on asynchronous settings, where agents move autonomously and independently (see [47, 48, 79]).

Summarizing, in the mobile agent model considered in this thesis, the agents have distinct id, they start from a single *homebase*, they communicate through *whiteboards*, and they are *synchronized*.

The Network. In this thesis the agents operate in two topologies: the *mesh* and the *torus*.

A $m \times n-$mesh $(m > 0; n > 0)$ has m rows and n columns. We denote by $r_0 \dots r_{m-1}$ the rows and by $c_0, , \dots, c_{n-1}$ the columns. A node u at row $r_x (0 \leq x < m)$ and column $c_y (0 \leq y < n)$ will be indicated by $P(x, y)$. All nodes are identical except the one initially containing an agent, which are the *homebase*.

A $m \times n-$torus network is similar to $m \times n-$mesh, except in the connection between the first and the last nodes in each dimension. These connections make all nodes connected with four neighbours SOUTH ($\Downarrow$), *EAST* ($\Rightarrow$), *NORTH* ($\Uparrow$) *and WEST* ($\Leftarrow$).

Each node contains a *whiteboard*, with a finite amount of memory (at most $\log N$ bits are sufficient, where $N = n \times m$ is the number of nodes in the network). The whiteboard is used for agent communication, and the access to the whiteboard is assumed to occur in a mutually exclusive manner. Each node has distinct ports leading to its neighbours. We assume the ports are consistently labelled SOUTH ($\Downarrow$), *EAST* ($\Rightarrow$), *NORTH* ($\Uparrow$) *and WEST* ($\Leftarrow$), and are visible to the agents.

The Problem and the Complexity Measures. We assume that the network is initially fully contaminated. The decontamination problem is that of cleaning (disinfecting) the entire network using a team of mobile antiviral systems agents injected into the system called cleaners. A cleaner can decontaminate any infected node it visits leaving the node immune to recontamination for a certain amount of time, called the *immunity time* of the system. However, once the cleaner departs, the decontaminated node can be re-contaminated by any contaminated neighbour once the immunity time elapses.

In this context, the efficiency of an algorithm is evaluated in terms of *number of agents* that are employed to perform the decontamination task.

The work with a focus closest to the one of the thesis are [55, 47, 48, 56]. The only comparable work (i.e., that considers exactly the same problem, in the same model) is [56], where the concept of temporal immunity has been introduced. The paper focus on trees only and shows optimal strategies for decontaminating trees. The techniques are however very specific to the tree topology and cannot be generalized to other network structures. The contributions of this thesis represent the first results of decontamination with temporal immunity by mobile agents in meshes and chordal rings.

3.3 The Cellular Automata Model

We also consider the problem in a weaker setting; we describe the global disinfection process by a set of cellular automata local rules. We consider the mesh, which is naturally described by a finite two-dimensional cellular automata, and torus, which is a circular cellular automata.

A two-dimensional cellular automata (CA) can be described by a quadruple $C = \langle \mathbb{Z}^2, \mathcal{S}, N, f \rangle$ where $\mathbb{Z}^2$ represents the set of *cells* (also called *sites* or *nodes*), $\mathcal{S}$ is the set of *states* of the cells, N is the *neighbourhood* of a cell, and $f : \mathcal{S}^{|N|} \to \mathcal{S}$ is the *local transition rule* (or simply *local rule*) of the automata.

A *finite* two-dimensional Boolean cellular automaton has a finite number of non-zero states in an infinite quiescent background, meaning, $C^t(z) = 0$ for all but finite $z \in \mathbb{Z}^d$. In this case, let $n \times n$ be the size of the finite lattice initially containing non-zero states. Let us indicate as (i, j) a cell in column i, row j with respect to the finite lattice indicating the left-bottom corner with $(0, 0)$. *Border* cells are cells $(i, n-1), (0, j), (i, 0), (n-1, j)$ with $0 < i, j < n-1$, the four *corner* cells are $(0, 0), (0, n-1), (n-1, 0), (n-1, n-1)$, and all other cells are *internal*. A *Circular* cellular automata can be thought of as a finite circular two-dimensional grid (or a torus). We will consider two types of neighbourhoods:

Von Neumann and *Moore*. Given a cell (i, j), the *von Neumann* neighbourhood consists of the cell itself, plus the four cells $(i, j-1), (i, j+1), (i-1, j), (i+1, j)$ at distance one in the Manhattan norm. The *Moore* neighbourhood, besides considering the cells of the Von Neumann neighbourhood, also includes the four neighbouring "diagonal cells" $(i-1, j-1), (i-1, j+1), (i+1, j-1), (i+1, j+1)$.

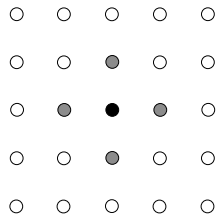

Figure 3.1: Cellular Automata with Von Neumann Neighborhood

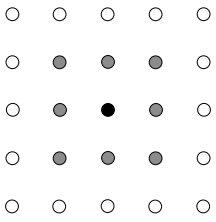

Figure 3.2: Cellular Automata with Moore Neighborhood

The Problem and the Complexity Measures. Like in the case of mobile agents, we assume that the lattice space is initially contaminated by a virus. In this case, however, we assume that the cells have local access to an antiviral software that can be activated to perform local decontamination leaving the cell decontaminated. When a cell is decontaminated, there is no guarantee it will not be contaminated again. When the immunity time expires the cell remains unprotected and becomes contaminated if any neighbour is contaminated. The cells cannot detect the presence of a virus in themselves and in their neighbourhood, and they cannot control the spread of contamination. They can however detect whether the antiviral process is active or has been activated in their neighbourhood. Moreover, the cells want to sparingly activate the anti-virus because when active it interrupts any other local computation (e.g., the trivial solution of simultaneously running the anti-virus in all cells would be unfeasible as it would completely disrupt the computation).

In the context of decontamination, the state space of the cellular automata consists of three states: $\{1, 0, \bullet\}$ *contaminated*; *decontaminated* (also referred to as *clean*); and *decontaminating* respectively. Given an initial configuration C^0, that is a mapping $C^0 : \mathbb{Z}^2 \to \{0, 1, \bullet\}$, cell states are synchronously updated at each time step by the local transition rule applied to their neighbourhoods. A configuration is the resulting map $C^t : \mathbb{Z}^2 \to \{0, 1, \bullet\}$ at any time t. When considering finite CA we will assume that "outside" the mesh everything is clean (e.g., the left neighbour of a node in the first column is considered *clean*).

The goal is to devise a strategy, for a given immunity time, that simultaneously decontaminates all cells, minimizing the number of cells simultaneously running the anti-virus at any time (i.e., the ones that are simultaneously *decontaminating*). Conversely, we are interested in minimizing the immunity time for a given maximum number of simultaneously decontaminating cells. The efficiency of an algorithm is then evaluated in terms of the *number of simultaneously decontaminating cells* for a given immunity time, or immunity time for a given maximum number of *decontaminating* cells.

Let $x^t_{i,j}$ denote the state of cell (i,j) at time t, and $N^t(i,j)$ the states of the neighbouring cells of (i,j) and that of its own state at time t (e.g., $N^t(i,j) = x^t_{i,j}, x^t_{i-1,j}, x^t_{i,j+1}, x^t_{i+1,j}$, $x^t_{i.j-1}$, in the case of Von Neumann neighbourhood).

Let s be the immunity time of a cell i in a *decontaminated* state at time t. When needed, we shall indicate such a time in parenthesis as follows: $x^t_{i,j} = 0(s)$.

The behavior of the system from time t to time $t+1$ can be then described by the following:

1. $x^{t+1}_{i,j} = f(N^t(i,j))$
2. If $x^{t+1}_{i,j} = x^t_{i,j} = 0$. Let $x^t_{i,j} = 0(s)$.
   If $s > 0$ then: $x^{t+1}_{i,j} = 0(s-1)$.
   If $s = 0$ and $((x^t_{i-1,j} = 1) \vee (x^t_{i+1,j} = 1) \vee (x^t_{i,j-1} = 1) \vee (x^t_{i,j+1} = 1))$ then: $x^{t+1}_{i,j} = 1$.

The first point indicates the change of state of a cell following the local rule of the cellular automaton. The second point indicates the degradation of the decontamination state; the node becomes less and less immune to contamination and eventually becomes contaminated if any one of the neighbouring cells are contaminated.

In this thesis we will also denote a transition rule by indicating the neighbouring states in clockwise order starting from the left neighbour. For example, in the case of Von Neumann neighbourhoods we will use the following notation to indicate a rule applied to cell $(i.j)$: $(x^t_{i,j}, x^t_{i-1,j}, x^t_{i,j+1}, x^t_{i+1,j}, x^t_{i.j-1}) \rightarrow x^{t+1}_{i,j}$. We will denote an arbitrary state by an asterisk $(*)$.

This thesis is the first where this particular problem is studied in the weak scenario of cellular automata. In [71, 105, 104] a similar model was employed to study the spread of contamination by cellular automata rules, but, in most cases, no decontamination was performed. When it was performed (e.g., [12, 54, 102]) no temporal immunity was ever assumed.

3.4 The Mobile Cellular Automata Model

This setting can be seen as a combination of the previous two models. It is a particular instance of models variously called *turmites*, *ants*, or *mobile cellular automata* [15, 67, 90, 20, 108, 9] where active elements move from node to node following local rules. Those models differ in the action taken when two active elements collide, which is usually probabilistic in the case of ants/turmites, while it is always deterministic in this setting.

In Mobile Cellular Automata (MCA), a team of active mobile entities (here called *agents*) move on a two-dimensional cellular space. Like in the case of Cellular Automata, a cell of the space is in a state belonging to a finite set and a cell changes its state according to the states of its neighbours. Unlike Cellular Automata, however, in MCA there is a special *active* state which corresponds to the presence of an agent; an active cell can not only change, its own state, but also the states of its neighbours. As for the case of CA, we will be considering finite cellular automata (with backgrounds of clean cells) and circular cellular automata, with both *Von Neumann* and *Moore* neighbourhoods at distance one.

As in Celluar Automata, state $x^t_{i,j} = 0$ corresponds to a *decontaminated* cell, state $x^t_{i,j} = 1$ to a *contaminated* cell. In this case, however, $x^t_{i,j} = \bullet$ has a different meaning and corresponds to an active cell containing an agent (i.e., *active* state).

The Problem and the Complexity Measures. Contamination occurs exactly like in the case of CA. Once decontaminated a cell is immune for T times units, and after time immunity is elapsed if one of its neighbours is contaminated it will become contaminated. On the other hand, decontamination is "performed" by the agents. In other words, a cell can change state *only by the action of an agent*. For example, the movement of an agent from cell (i, j) to cell $(i + 1, j)$ at time t is described by the change of state of both cells: let $x^t_{i,j}$ be in state *decontaminating* indicating the presence of an agent at time t in cell (i, j), for the movement to take place, $x^{t+1}_{i,j}$ becomes *decontaminated* , while $x^{t+1}_{i+1,j}$ becomes *decontaminating*. The system is updated synchronously at discrete time steps. In this case, however, the local rule applies only to active cells and returns a new state for the cell *and* a movement direction indicating to which neighbours the agent is moving. A cell in any state receiving an agent becomes decontaminating.

A distinction has to be made regarding the possibility of an active cell to create copies of itself (*cloning*). If the system has cloning capabilities, the output of the local function can contain any number of directions and the active cell itself can stay active. If instead

the system does not have cloning capabilities, the local rule returns a pair containing the new state and a single direction of movement. We will denote the transition rule by f: the rule takes as input a set of states and returns a pair containing its own state and a set of directions. For example, the rule f applied to cell $(i.j)$ is indicated as follows: $f(N^t(i,j)) = (x_{i,j}^{t+1}, mov)$ where $N^t(i,j)$ indicates the states of the neighbouring cells of (i,j) and that of its own state at time t, and mov is a set of directions from $\{\Uparrow, \Downarrow, \Leftarrow, \Rightarrow\}$. Let $f_1()$ denote the first component $x_{i,j}$ of the output of the transition rule f, and $f_2()$ the second component mov. In the remainder of this work the transition rule for an active cell is indicated in a table where, for a given configuration, we show the next state of the cell and the direction of the agent's movement. A configuration always indicates the state of the cell itself, and its neighbourhood from the left neighbour in clockwise order (e.g., in the case of Von Neumann neighbourhood: $x_{i,j}^t, x_{i-1,j}^t, x_{i,j+1}^t, x_{i+1,j}^t, x_{i.j-1}^t$).

The behaviour of the system from time t to time $t+1$ can be then described by the following:

1. For all cells (i,j), regardless of their state $x_{i,j}^t$.
   $x_{i,j}^{t+1}$ becomes *active* if $((f_2(N^t(i-1,j)) = \Rightarrow) \vee (f_2(N^t(i,j+1)) = \Downarrow) \vee (f_2(N^t(i+1,j)) = \Leftarrow) \vee (f_2(N^t(i,j+1)) = \Uparrow))$.

2. If 1) does not apply and $x_{i,j}^t$ is *active*.
   $x_{i,j}^{t+1} = f_1(N^t(i,j))$

3. If 1) does not apply and $x_{i,j}^t = 0(s)$.
   If $s > 0$ then: $x_{i,j}^{t+1} = 0(s-1)$.
   If $s = 0$ and $((x_{i-1,j}^t = 1) \vee (x_{i+1,j}^t = 1) \vee (x_{i,j-1}^t = 1) \vee (x_{i,j+1}^t = 1))$ then: $x_{i,j}^{t+1} = 1$.

An important observation has to be made regarding the behaviour of the system when two or more agents move into the same cell. The rules described above transform a cell that receives at least one agent to *active*, which means that if more than one agent move into the same cell they are fused into a single agent. The number of agents in the system might then decrease over time.

We will consider *cloning* agents that can create copies of themselves, in which case a cell in state $\bullet$ could propagate in several direction. We will also consider non-cloning agents that cannot duplicate. In the case of non-cloning agents, the agent moves (like a physical agent) in a single direction.

Given a MCA of size $n \times n$ and an immunity time T, our goal is to choose the initial location of the *active* cells and the local transition rule f to achieve global decontamina-

tion in such a way that the number of active cells at a given time is as low as possible. The efficiency of an algorithm is evaluated in terms of the *number of agents* that are employed to perform the decontamination task.

Unlike the mobile agents setting, in this model the agents do not have communication power nor memory. They do however have visibility capability (i.e., they can observe the state of their neighbours). Note that in both MCA and MA the agents are the active elements of the system and have the capability to change the state of a cell from *contaminated* to *clean*. Agents are *anonymous*, and can start the decontamination from different location *homabases* on the first column. The cells themselves, on the other hand, are passive in the sense that they cannot change their state from *contaminated* to *clean* without the presence of an agent. This is not the case in CA, where the cells themselves are the active elements.

The example below shows the different way in which agents move in this model compared to the movements they can perform in the MA model. Suppose we have a fully contaminated MCA in a Moore neighbourhood with an agent on the upper left corner. We want to define the local rules that allow the agent to move down in the first column while decontaminating its nodes, then to turn right and stop. The down-movement is obtained by rule $(\bullet, 0, 0, 0, 0, 1, 1, 1, 0) \rightarrow (0, \Downarrow)$ which makes the current cell become *decontaminated* if its right, bottom, and diagonal-right-bottom neighbours are *contaminated*, and lets the agent move down. The movement to the right at the end of the column is obtained by rule $(\bullet, 0, 0, 0, 1, 1, 0, 0, 0) \rightarrow (0, \Rightarrow)$. The behaviour shown in Figure 3.3 is obtained by the set of simple rules in Table 3.1.

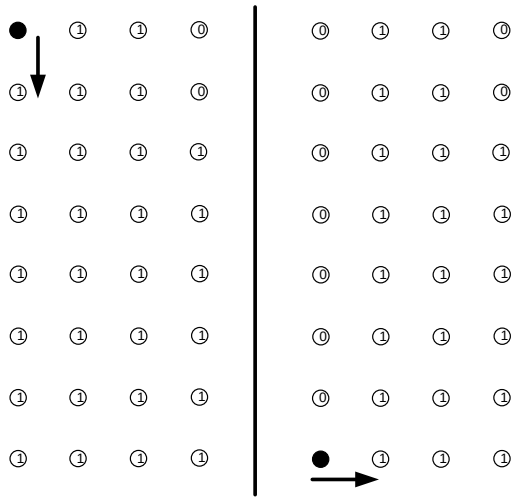

Figure 3.3: Example of transition with temporal decontamination in a finite mobile cellular automata

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 0, 0, 1, 1, 1, 0\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 1, 1, 0, 0, 0\}$	0	$\Rightarrow$

Table 3.1: Example of rules of transition with temporal decontamination in a finite mobile cellular automata

Agents used in this model are similar to the oblivious agents used in [50, 49, 79, 78], which have never been employed in the context of decontamination. The closest model used in this context is the one in [5, 47, 48] where, in contrast to MCA, the agents also have unbounded memory of the past. This thesis contains the first results for the decontamination problem in this setting.

Chapter 4

Decontamination by Cellular Automata

In this chapter, we look at decontamination by cellular automata. We describe the global decontamination process by a set of cellular automata local rules. We consider the mesh (or grid), which is naturally described by a finite two-dimensional, three state cellular automata. we devise decontamination rules both in the common situation where after being decontaminated a cell is prone to re-contamination by contact, and in a setting where decontamination leaves the cells immune to recontamination for a certain amount of time (temporal immunity). We also distinguish between Von Neumann and Moore neighbourhoods, showing that not surprisingly, a bigger neighbourhood allows for a more efficient decontamination in the temporal case. In the case of basic decontamination, if we increase the neighbourhood to include diagonal neighbours, we cannot obtain a better result.

In [26] results obtained in this chapter for: basic decontamination in the case of Von Neumann neighbourhood, Temporal decontamination in the case of Von Neumann neighbourhood with $k = 1, 2, 4$ simultaneously decontaminating cells, Temporal decontamination in the case of Moore neighbourhood with a single and multiple simultaneously decontaminating cells, are published.

4.1 Basic Decontamination

In this section we consider basic decontamination in finite and circular two-dimensional cellular automata, where the decontaminating process does not provide any type of im-

munity (e.g., immunity time $t = 1$) and a cell could be contaminated as soon as one of its neighbours is contaminated. Any decontamination strategy has to forbid a *decontaminated* cell to ever have contact with a *contaminated* one, which can propagate contamination to the *decontaminated* cell. In the case of basic decontamination, increasing the neighbourhood to include diagonal neighbours, does not obtain better results. In fact, the lower bound on the number of simultaneously *decontaminating* cells holds both for finite and for circular cellular automata. As we will see later, when considering temporal decontamination, the type of neighbourhood has an impact on the efficiency of the solution.

4.1.1 Finite Cellular Automata with the Von Neumann Neighbourhood

With basic decontamination, a single decontaminating cell per time unit would obviously not be sufficient to perform decontamination in a finite two-dimensional cellular automata because immediately after becoming *decontaminated* a cell would inevitably be exposed to a *contaminated* cell as at most one of its neighbours could become *decontaminating*. The question is what is the minimum number of decontaminating cells which could guarantee decontamination without recontamination in a finite two-dimensional cellular automata. We prove in the following that n decontaminating cells are necessary and sufficient.

Theorem 4.1. *Optimal basic decontamination can be achieved in a finite two-dimensional cellular automata with the Von Neumann neighbourhood using n simultaneous decontaminating cells.*

Proof. To prove that n simultaneous decontaminating cells are necessary and sufficient, we need to prove that a set of rules exists. Starting with n *decontaminating* cells initially located at the first column, all cells will become decontaminated by the end of the process, and no cell will be recontaminated. We also need to prove that there is no other solution that achieves the decontamination with less than n *decontaminating* cells.

The set of rules is described in Table 4.1 (all missing combinations of states leave the cell unchanged). The idea is to place the initial decontaminating cells in each cell of the first column of the cellular automata and to have the cells act according to the following very simple rules (see Table 4.1). Regardless of the state of the neighbouring cells, a decontaminating cell becomes decontaminated. A decontaminated cell becomes decontaminating at time $t + 1$ if its left neighbour is in a *decontaminating* state at time

t. The effect of the local rules is the sequential decontamination of columns, where a decontaminated cell is obviously never in contact with a contaminated one. The cellular automata is then fully decontaminated in n time units (see Figure 4.1).

To prove that n simultaneous decontaminating cells are necessary, consider a time t during the decontamination process when there are h *decontaminated* cells. To avoid recontamination at time t, these cells must be surrounded by *decontaminating* cells (or by quiescent cells outside the border of the cellular automata); we will call such cells *protective cells*. If the *decontaminated* cells form separate continuous blocks, the amount of necessary protective cells is never smaller than if they belonged to a single block. It has been shown in [106] that for a block of size h at least $\min\{n, \lfloor\frac{1+\sqrt{1+8h}}{2}\rfloor\}$ protective cells are necessary. Since $\lfloor\frac{1+\sqrt{1+8h}}{2}\rfloor < n$ implies $h < \frac{n(n-1)}{2}$, we have that less than n simultaneous *decontaminating* cells can protect less than half the cells of the cellular automata, which then cannot be fully decontaminated. □

Configuration	Next State
$\{\bullet, *, *, *, *\}$	0
$\{1, \bullet, 0, 1, 1\}$	$\bullet$
$\{1, \bullet, 0, 0, 1\}$	$\bullet$
$\{1, \bullet, 1, 1, 1\}$	$\bullet$
$\{1, \bullet, 1, 1, 0\}$	$\bullet$
$\{1, \bullet, 1, 0, 1\}$	$\bullet$
$\{1, \bullet, 1, 0, 0\}$	$\bullet$

Table 4.1: Rule Table for Basic Decontamination in Finite Cellular Automata with the Von Neumann neighbourhood.

4.1.2 Circular Cellular Automata the with Von Neumann Neighbourhood

A similar idea achieves optimal decontamination in a circular cellular automata. The initial *decontaminating* cells are placed in any two consecutive columns of the cellular automata and the cells obey the following simple rules: regardless of the neighbouring cell's state, a *decontaminating* cell becomes decontaminated; a *contaminated* cell becomes *decontaminating* at time $t+1$ if its left cell, its right cell, or both are *decontaminating* at

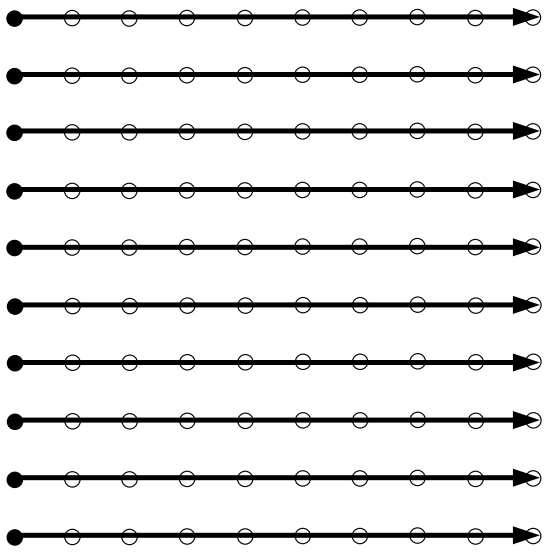

Figure 4.1: Basic Decontamination in Finite two-dimensional Cellular Automata with the Von Neumann neighbourhood

time t. The set of rules is shown in Table 4.2 (all missing combinations of states leave the cell unchanged).

Configuration	Next State
$\{\bullet, *, *, *, *\}$	0
$\{1, \bullet, 1, 1, 1\}$	$\bullet$
$\{1, 1, 1, \bullet, 1\}$	$\bullet$
$\{1, \bullet, 1, \bullet, 1\}$	$\bullet$

Table 4.2: Rule Table for Basic decontamination in Circular two-dimensional Cellular Automata with the Von Neumann Neighbourhood

The effect of the local rules is the sequential decontamination of pairs of columns, where a decontaminated cell is never in contact with a contaminated one. The cellular automata is then fully decontaminated in $\lfloor \frac{n}{2} \rfloor$ time units (see Figure 4.2).

Theorem 4.2. *Optimal basic decontamination can be achieved in a circular cellular automata with the Von Neumann neighbourhood using* $2n$ *simultaneous decontaminating*

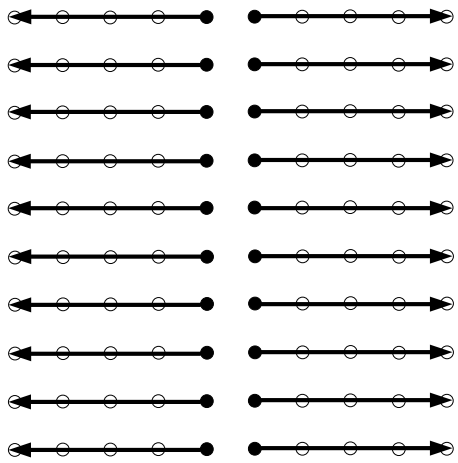

Figure 4.2: Basic Decontamination in Circular Cellular Automata with the Von Neumann neighbourhood

cells.

Proof. To prove that $2n$ simultaneous decontaminating cells are sufficient, we need to prove that starting with $2n$ decontaminating cells initially located at two consecutive columns, all cells will become decontaminated by the end of the process, and no cell will be re-contaminated. That is, once decontaminated, every cell stays decontaminated until the end of the process, when all cells are decontaminated. We can prove by induction on the number of columns that: initially decontaminating cells are located in columns (i, j) and $(i, j + 1)$: (i) if n is even there is a time when all cells in columns $(i, j + \frac{n}{2} - 1)$ and $(i, j - \frac{n}{2} - 1)$ are in *decontaminated* state and cells in columns $(i, j + \frac{n}{2})$ and $(i, j - \frac{n}{2})$ are in a *decontaminating* state, and (ii) if n is odd there is a time when all cells in columns $(i, j + \lfloor \frac{n}{2} \rfloor - 1)$ and $(i, j - \lfloor \frac{n}{2} \rfloor - 1)$ are in a *decontaminated* state and cells in column $(i, j + \lceil \frac{n}{2} \rceil)$ are in a *decontaminating* state.

As for optimality, we now show that it is not possible to perform basic decontamination in a circular cellular automata with less than $2n$ simultaneously *decontaminating* cells. First, notice that following a simple geometric reasoning the number of agents k needed to cover the perimeters of the *decontaminated* blocks is greater than or equal to the number that would be required if these blocks were to be attached (i.e., forming a single block). To minimize the number of *decontaminating* cells, at any point in time the *decontaminated* cells must be connected. Second, it is easy to show that the perimeter of a single block (i.e., the *decontaminated* nodes of the block in contact with *contaminated*

ones) is minimized when it is as close as possible to a rhombus as shown in Figure 4.3.

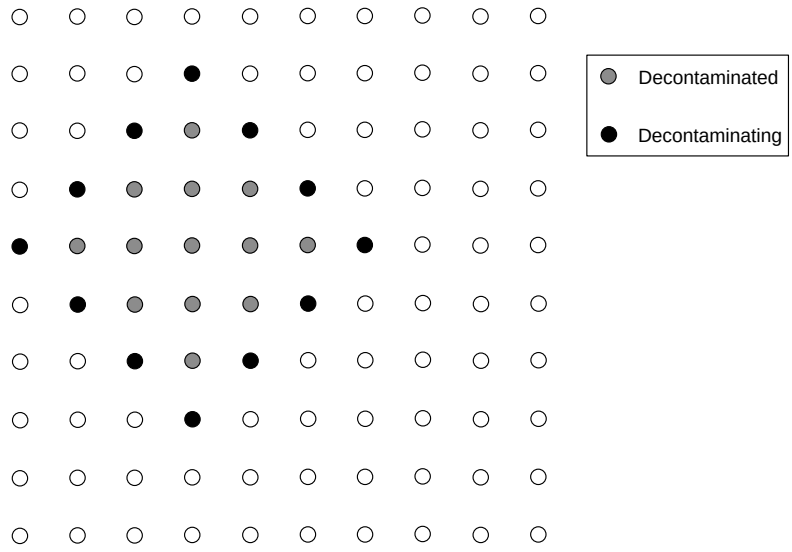

Figure 4.3: Block, for a given area which has a minimum perimeter.

Let us now compute the number of simultaneously decontaminating cells k needed to cover the perimeter of the rhombus. For any other possible shape, the number of decontaminating cells k will be greater than k. Given a rhombus of side l, let f be the number of *decontaminated* or *decontaminating* cells. The perimeter of the rhombus is $4\sqrt{f}-4$, and the area (composed of *decontaminated* or *decontaminating* cells) is: $f = l^2 + (l-1)^2$. Thus, the number of simultaneously *decontaminating* cells k needed to cover the perimeter of any shape is $4\sqrt{f}-4$. From $f = l^2+(l-1)^2$ we derive that $l = \frac{1+\sqrt{2f-1}}{2}$, substituting l we have; $k \geq 2 \times \sqrt{2f-1} - 2$. Consider a moment during the decontamination when there are $f = l^2 + (l-1)^2 \geq \frac{n^2+2n+2}{2}$ decontaminated cells. This holds as long as $n \times n \geq \frac{n^2+2n+2}{2}$, i.e., $n \geq 3$. From $f = l^2 + (l-1)^2 \geq \frac{n^2+2n+2}{2}$ it follows that the number of required simultaneously decontaminating cells is $\geq 2\sqrt{2 \times (\frac{n^2+2n+2}{2}) - 1} - 2 = 2n$, thus, the number of required simultaneously decontaminated cell is at least $2n$. □

4.1.3 Circular Cellular Automata with the Moore Neighbourhood

In the case of basic decontamination, it is easy to see that if we increase the neighbourhood to include diagonal neighbours, we cannot obtain a better result. In fact, the lower bound on the number of simultaneously *decontaminating* cells also holds in this case

both for finite and for circular cellular automata. As we will see later, when considering temporal decontamination, the type of neighbourhood has instead an impact on the efficiency of the solution.

4.2 Temporal Decontamination

The case of temporal decontamination is quite different and more interesting. With temporal decontamination, once a cell becomes *decontaminated*, it stays decontaminated for a certain amount of time ($t > 1$), called immunity time. We must design a set of local rules and choose the location of the initial *decontaminating* cells in such a way that during the evolution of the cellular automata, a *decontaminated* cell never comes into contact with a *contaminated* one after its immunity time has expired. We distinguish here the two types of neighbourhoods: Von Neumann and Moore, noticing that the amount of influence from neighbouring cells has a large impacts on the efficiency solutions.

4.2.1 Finite Cellular Automata

We describe the solutions that achieve decontamination in the case of finite cellular automata with temporal immunity with Von Neumann and Moore neighbourhoods. In the following, and in the rest of this chapter, when we say that decontamination "propagates", we indicate that a *decontaminating* cell becomes *decontaminated* and one or more of its neighbours become *decontaminating*, thus simulating the propagation of decontamination from cell to neighbouring cell(s).

4.2.1.1 Finite Cellular Automata with the Von Neumann Neighbourhood

We first show that if we impose the use of a single *decontaminating* cell per time unit, temporal decontamination can be achieved if and only if the immunity time is at least $4(n-1)-1$.

Theorem 4.3. *With a single decontaminating cell per time unit, temporal decontamination is possible if and only if the immunity time is* $\geq 4(n-1)-1$.

Proof. Decontamination is achieved by placing the initial decontamination cell at a corner and by applying the set of rules indicated in Table 4.3. The effect of the local rules is the spiral propagation of the decontaminating cell (see Figure 4.4) where a *decontaminated* cell is never in contact with a *decontaminated* one for more than $4(n-1)-1$ time units,

Configuration	Next State
$\{\bullet, *, *, *, *\}$	0
$\{1, \bullet, 1, 1, 0\}$	$\bullet$
$\{1, \bullet, 1, 0, 0\}$	$\bullet$
$\{1, 0, \bullet, 1, 1\}$	$\bullet$
$\{1, 0, \bullet, 1, 0\}$	$\bullet$
$\{1, 0, \bullet, 0, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 1\}$	$\bullet$
$\{1, 1, 0, \bullet, 1\}$	$\bullet$
$\{1, 1, 1, 0, \bullet\}$	$\bullet$
$\{1, 1, 0, 0, \bullet\}$	$\bullet$
$\{1, 0, 0, 0, \bullet\}$	$\bullet$

Table 4.3: Rule Table for Temporal Decontamination with Single Decontaminating cell in Finite Cellular Automata with the Von Neumann Neighbourhood.

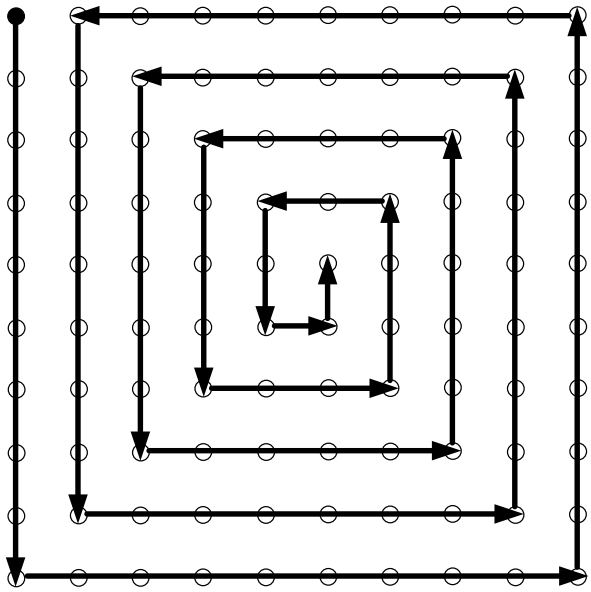

Figure 4.4: Propagation of Single Decontaminating cell with Temporal Immunity and the Von Neumann neighbourhood

resulting in a situation where all cells are simultaneously decontaminated after $n \times n$ time units.

We now show that there is no initial placement for which a correct set of rules exists when the immunity time is smaller than $4(n-1)-1$. A correct set of rules is such that it maintains a single *decontaminating* cell per time unit, and eventually decontaminates the whole network never leaving a *decontaminated* cell in contact with a *contaminated* one for more than $4(n-1)-1$ time units after it has been decontaminated. We have three possible placements of the initial decontaminating cell: corner $((0,0),(0,n-1),(n-1,0),(n-1,n-1))$; border $((0,j)$ or $(n-1,j)$ with $0<j<n-1$, $(i,0)$, or $(i,n-1)$ with $0<1<n-1)$; and internal $((i,j)$ with $0<i,j<n-1)$.

Case 1- Internal: Let us first consider the case when the initial *decontaminating* cell is internal. In this case we will show that, regardless of the immunity time, there exists no set of rules that allows for the maintenance a single decontaminating cell. The other cases follow a similar argument. Let us call a rule that cannot be present in a correct set of rules a *forbidden rule*. We now construct all possible dynamics by forbidding impossible behaviour. Notice that $r_0 = \{\bullet, *, *, *, *\} \to 0$ is necessarily part of the set of rules to ensure the presence of a single decontaminating cell. Moreover, to allow the spread of decontamination to a single new cell, one (and only one) of the following four rules must be present: $r_1 = \{1, \bullet, 1, 1, 1\} \to \bullet$; $r_2 = \{1, 1, \bullet, 1, 1\} \to \bullet$; $r_3 = \{1, 1, 1, \bullet, 1\} \to \bullet$; or $r_4 = \{1, 1, 1, 1, \bullet\} \to \bullet$.

Without loss of generality, we can choose any rule because any other choice will produce a symmetric propagation in the cellular automata. Let r_1 be present and let r_2, r_3, r_4 be forbidden. The combination of r_0 and r_1 makes the decontaminating cell propagate to the right until reaching the border. At this point, for ensuring the propagation of the decontaminating cell, one (and only one) of the following three rules must be added to the set: $r_5 = \{0, 0, 1, \bullet, 1\} \to \bullet$; $r_6 = \{1, 1, 1, 0, \bullet\} \to \bullet$; and $r_7 = \{1, 1, \bullet, 0, 1\} \to \bullet$. We can show that r_5 is forbidden. The decontaminating cell would propagate back to the original cell where because of the fact that r_2, r_3, and r_4 are forbidden, and the only possible movement would be to the right again giving rise to a cycle. Without loss of generality, because rules r_6 and r_7 will produce a symmetric behaviour, let r_6 be present an r_7 be forbidden. Because of r_6 being present, rule $r_8 = \{1, 1, 1, \bullet, 0\} \to \bullet$ must be forbidden, otherwise more than one cell would be in a *decontaminating* state. The combination of r_0 and r_6 makes the *decontaminating* cell propagate up until reaching the lower neighbour of the corner. Since r_3 is forbidden, we need to add a rule to ensure propagation. The only possible rule is $r_9 = \{1, 1, 0, 0, \bullet\} \to \bullet$ and the decontamination will reach the

corner. At this point, to ensure the propagation of the decontaminating cell, one (and only one) of the following two rules must be added to the set: $r_{10} = \{0, 1, \bullet, 0, 0\} \rightarrow \bullet$), or $r_{11} = \{1, 1, 0, \bullet, 1\} \rightarrow \bullet$. Rule r_{10} is forbidden because the decontaminating cell would propagate back to the the first decontaminated cell on the border and only a cycle is permitted (all the other rules are forbidden). Following a similar reasoning, one can see that the decontaminating cell is forced to propagate left until reaching the corner with the necessary inclusion of rule $r_{12} = \{1, 0, 0, \bullet, 1\} \rightarrow \bullet$, then down following another necessary rule $r_{14} = \{1, 0, \bullet, 1, 1\} \rightarrow \bullet$. At the next step, however two cells ($(0, n-3)$ and $(1, n-2)$) will become simultaneously decontaminating (due to rules r_1 and r_{14} respectively), which contradicts our hypothesis. It follows that starting from an internal cell we cannot decontaminate all cells with a single *decontaminating* cell per time unit.

Case 2- Border: Let us consider the case when the initial decontaminating cell is a border cell (i.e., cell (i, j), with $((i, j) \neq (0, n-1), (n-1, 0), (0, 0), (n-1, n-1))$. Without loss of generality, let us consider a cell on the left border, any other choice of a border cell will results in a symmetric behaviour. Notice that $r_0 = \{\bullet, *, *, *, *\} \rightarrow 0$ is necessarily part of the set of rules to ensure the presence of a single decontaminating cell. To allow the spread of decontamination to a single new cell, one (and only one) of the following three rules must be present: $r_1 = \{1, 0, \bullet, 1, 1\} \rightarrow \bullet$; $r_2 = \{1, 0, 1, 1, \bullet\} \rightarrow \bullet$; or $r_3 = \{1, \bullet, 1, 1, 1\} \rightarrow \bullet$.

Without loss of generality, let r_1 be present and let r_2 and r_3 be forbidden. However, the combination of r_0 and r_3 makes the decontaminating cell propagate to the internal cells similar to the previous placements (starting from an internal cell), combination of r_2 and r_0 will produce a symmetric behaviour. The combination of r_0 and r_1 propagates down until reaching the upper neighbours of the lower border, To ensure the propagation of the *decontaminating* cell, the only possible rule to add to the set of rules is $r_7 = \{1, 0, \bullet, 1, 0\} \rightarrow \bullet$ (otherwise the decontamination will stop) which propagates the decontamination to the lower corner. At this point, only one rule is possible, $r_8 = \{1, \bullet, 1, 1, 0\} \rightarrow \bullet$ (otherwise the decontamination will stop). This rule propagates the decontaminating cell to the right since rule $r_2 = \{1, 0, 1, 1, \bullet\} \rightarrow \bullet$ is forbidden and the only possible propagation is the right propagation. Rule $r_9 = \{1, \bullet, 1, 0, 0\} \rightarrow \bullet$ is the only possible rule to add to the set of rules and propagate the *decontaminating* cell to the lower right corner. Only one rule is now possible $r_{10} = \{1, 1, 0, 0, \bullet\} \rightarrow \bullet$, which makes the *decontaminating* cell propagate up until reaching the lower neighbour of the upper right corner. Rule $r_{11} = \{1, 1, 0, 0, \bullet\} \rightarrow \bullet$ is the only possible rule to propagate the decontaminating cell to the upper right corner. At this time, rule $r_{12} = \{1, 1, 0, \bullet, 1\} \rightarrow \bullet$

propagates the decontaminating cell to the left, and $r_{13} = \{1, 0, 0, \bullet, 1\} \to \bullet$ propagates the *decontaminating* cell to the upper left corner (if the initial decontaminating cell is the lower neighbour of the corner the rule must be $r_{14} = \{1, 0, 0, \bullet, 0\} \to \bullet$), Rule $r_1 = \{1, 0, \bullet, 1, 1\} \to \bullet$ propagates down to reach the upper neighbour of the initial decontaminating cell. The distance traversed is $4(n-1)-1$, however, the left neighbour of the initial decontaminating cell is still contaminated. To decontaminate it at least two more time units are required, but another contradiction can be seen. To decontaminate the cells in the next column $r_{15} = r_3 = \{1, \bullet, 1, 1, 1\} \to \bullet$ must be added which is a forbidden rule.

Starting from a corner would be similar. In fact, starting from the corner would be the best strategy (as we know), but still cannot achieve $4(n-1)-2$). □

The strategy can be generalized when we allow two or four simultaneous decontaminating cells. In the first case, we achieve decontamination by initially placing the two decontaminating cells in two opposite corners and having the rules of Table 4.4 describe the local behaviour of the cells. In the second case, the four agents are initially placed in the corners and the rules are described in Table 4.5. The global behaviour of the automata corresponds to the propagation of the decontamination in spirals (see Figure 4.5, Figure 4.6) and with an analogous reasoning we can show that the immunity time is at least $2(n-1)-1$ for the case of two decontaminating cells, and $(n-1)-1$ for four. We can then conclude

Theorem 4.4. *With $k = 2, 4$ decontaminating cells per time unit, temporal decontamination is possible if and only if the immunity time is $\geq \frac{4}{k}(n-1)-1$.*

Configuration	Next State
$\{\bullet, *, *, *, *\}$	0
$\{1, \bullet, 1, 1, 0\}$	$\bullet$
$\{1, \bullet, 1, 0, 0\}$	$\bullet$
$\{1, 0, \bullet, 1, 1\}$	$\bullet$
$\{1, 0, \bullet, 1, 0\}$	$\bullet$
$\{1, 1, 0, \bullet, 1\}$	$\bullet$
$\{1, 0, 0, \bullet, 1\}$	$\bullet$
$\{1, 1, 1, 0, \bullet\}$	$\bullet$
$\{1, 1, 0, 0, \bullet\}$	$\bullet$
$\{1, \bullet, 0, 0, \bullet\}$	$\bullet$
$\{1, 0, \bullet, \bullet, 0\}$	$\bullet$
$\{1, 0, \bullet, 0, \bullet\}$	$\bullet$

Table 4.4: Rule Table for Temporal Decontamination with Two Decontaminating cells in Finite Cellular Automata with the Von Neumann Neighbourhood.

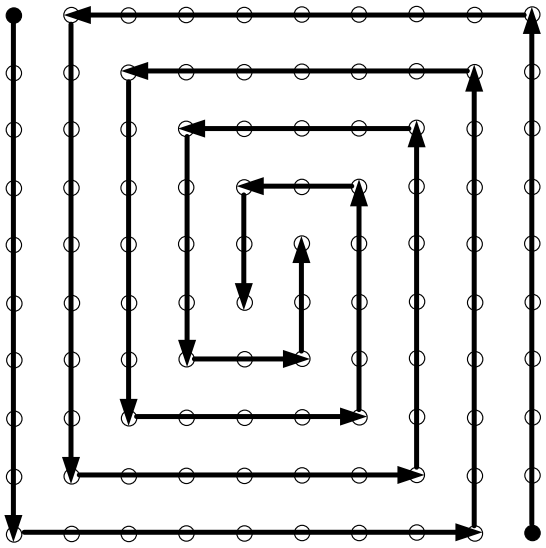

Figure 4.5: Propagation of two Decontaminating cells with Temporal Immunity and Von Neumann Neighbourhood

Configuration	Next State
$\{\bullet, *, *, *, *\}$	0
$\{1, \bullet, 1, 1, 0\}$	$\bullet$
$\{1, \bullet, 1, 0, 0\}$	$\bullet$
$\{1, 0, \bullet, 1, 1\}$	$\bullet$
$\{1, 0, \bullet, 1, 0\}$	$\bullet$
$\{1, 1, 0, \bullet, 1\}$	$\bullet$
$\{1, 0, 0, \bullet, 1\}$	$\bullet$
$\{1, 1, 1, 0, \bullet\}$	$\bullet$
$\{1, 1, 0, 0, \bullet\}$	$\bullet$
$\{1, \bullet, 1, \bullet, 0\}$	$\bullet$
$\{1, \bullet, 0, \bullet, 1\}$	$\bullet$
$\{1, 0, \bullet, 1, \bullet\}$	$\bullet$
$\{1, 1, \bullet, 0, \bullet\}$	$\bullet$
$\{1, \bullet, \bullet, \bullet, \bullet\}$	$\bullet$

Table 4.5: Rule Table for Temporal Decontamination with Four Decontaminating cells in Finite Cellular Automata with the Von Neumann Neighbourhood.

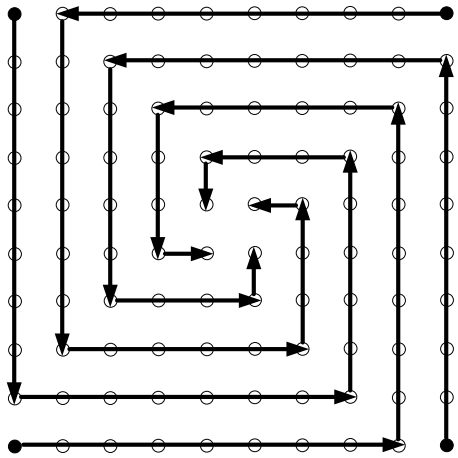

Figure 4.6: Propagation of Four Decontaminating cell with Temporal Immunity and the Von Neumann Neighbourhood

4.2.1.2 Finite Cellular Automata with the Moore Neighbourhood

The bounds are different if we increase the neighbourhood to include diagonal neighbours. With a single decontaminating cell optimality is reached when the immunity time is at

least $2n - 1$. We now describe a general set of rules which achieves optimality in the particular case of a single agent placed on a corner, but which also works correctly with more agents.

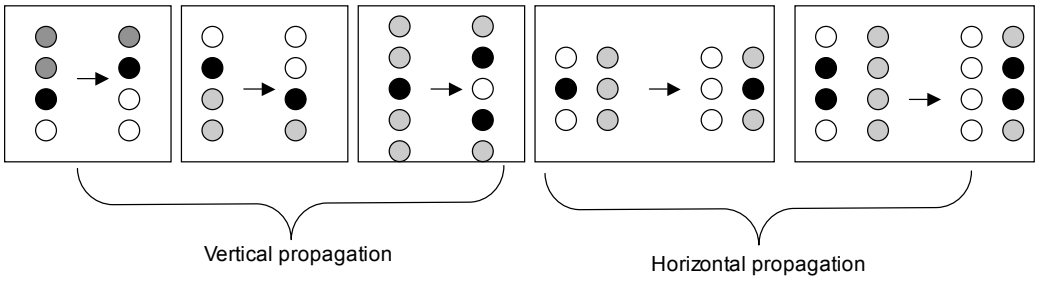

Figure 4.7: Propagation of Decontaminating cell with Temporal Immunity and the Moore Neighbourhood

Any number of *decontaminating* cells (greater than 1) are initially placed on the first column at distance greater than 1 from each other (i.e., no two consecutive cells are initially decontaminating). Careful inspection of the rules in Table 4.6 shows that decontamination "propagates" vertically to contaminated cells (a *contaminated* cell becomes *decontaminating* when its lower / upper neighbour, or both, are *decontaminating*) until they reach either another decontaminating cell or a clean one, in which case they "propagate" to the next column (a *contaminated* cell becomes *decontaminating* when its left neighbour is *decontaminating* and at least one of its left diagonal neighbours are *decontaminated*). Figure 4.7 shows the effect of the local rules on some partial configurations.

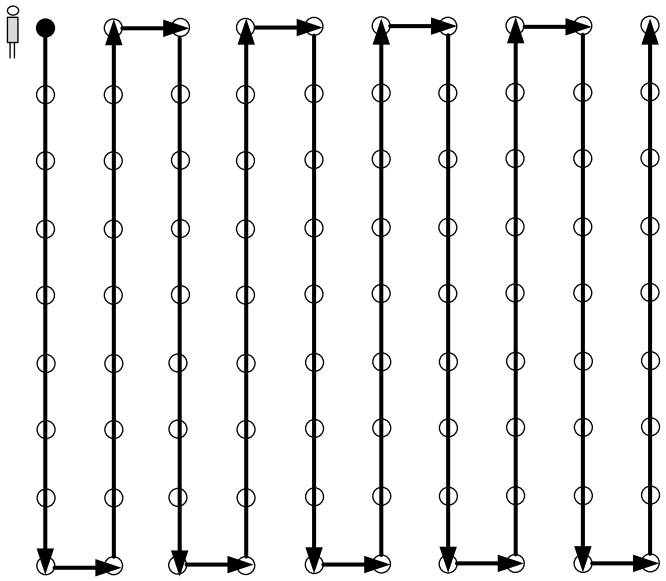

Figure 4.8: Propagation of Single Decontaminating cell with Temporal Immunity and Moore Neighborhood

Configuration	Next State
$\{\bullet, *, *, *, *, *, *, *, *\}$	0
$\{1, \bullet, 0, 1, 1, 1, 1, 1, 0\}$	$\bullet$
$\{1, \bullet, 0, 0, 0, 1, 1, 0, 0\}$	$\bullet$
$\{1, \bullet, 0, 1, 1, 1, 0, 0, 0\}$	$\bullet$
$\{1, \bullet, 0, 0, 0, 1, 1, 1, 0\}$	$\bullet$
$\{1, \bullet, 0, 1, 1, 1, 1, 1, \bullet\}$	$\bullet$
$\{1, \bullet, 0, 1, 0, 0, 0, 1, 0\}$	$\bullet$
$\{1, \bullet, 0, 1, 0, 0, 0, 0, 0\}$	$\bullet$
$\{1, \bullet, 0, 0, 0, 0, 0, 1, 0\}$	$\bullet$
$\{1, \bullet, \bullet, 1, 1, 1, 1, 1, 0\}$	$\bullet$
$\{1, \bullet, 0, 1, 0, 0, 0, 1, \bullet\}$	$\bullet$
$\{1, \bullet, \bullet, 1, 0, 0, 0, 1, 0\}$	$\bullet$
$\{1, \bullet, \bullet, 0, 0, 0, 0, 0, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 0, 0, 0, 0, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 1, 1, 1, 1, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 1, 1, 0, 0, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 0, 0, 0, 1, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 1, 1, 1, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 0, 0, 0, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, 1, 0, 0, 0, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, 0, 0, 0, 0, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, 0, 0, 1, 1, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, 1, 1, 1, 1, \bullet, 0\}$	$\bullet$

Table 4.6: Rule Table for Temporal Decontamination with Multiple Decontaminating Cells in Finite Cellular Automata with the Moore Neighbourhood.

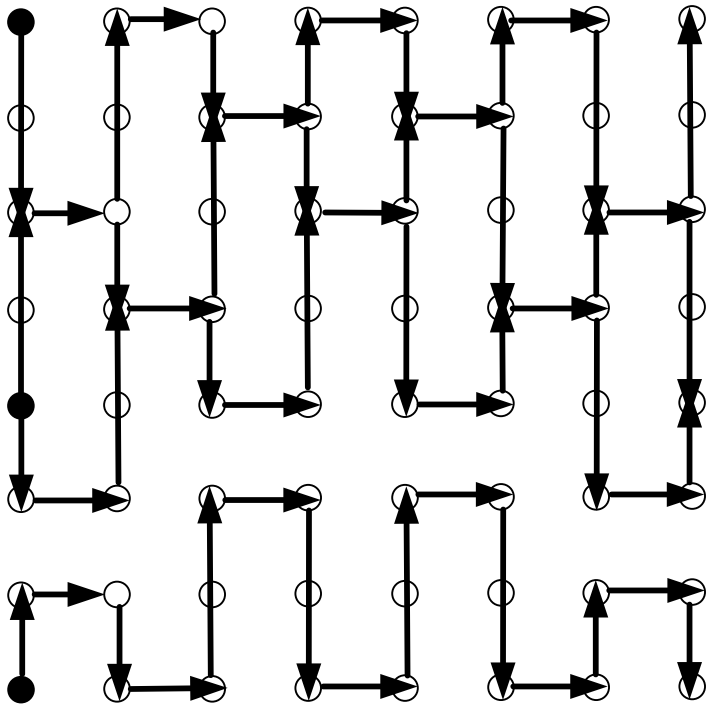

Figure 4.9: Propagation of Multiple Decontaminating cell with Temporal Immunity and the Moore Neighbourhood

Figure 4.8 shows the global propagation path of the decontamination in the case of a single decontaminating cell, while Figure 4.9 describes instead the global propagation path of the decontamination in the case of multiple decontaminating cells. In the following we prove the correctness of this set of rules.

Theorem 4.5. *Let d_{max} be the maximum distance between two consecutive* decontaminating *cells or between a* decontaminating *cell and a corner at time 0. If the initial number of decontaminating cells is at least* 2*, the set of rules in Table 4.6 achieves decontamination with immunity time $t = d_{max}$.*

Proof. To prove the theorem, we need to prove that decontamination is achieved monotonically. That is, once *decontaminated*, every cell stays clean until the end of the process, when all cells are *decontaminated*. Let us call *entry points* of decontamination in a column, the cells in such a column that become decontaminating due to a left decontaminating neighbour (horizontal propagation).

We now prove by induction on the number of columns that for each column i there is a time t when: (i) all cells in column i are either *decontaminated* or *decontaminating*; (ii) all cells in column $i-1$ (for $i > 0$) are decontaminated; (iii) by time $t + d_{max}$ all right neighbours of a *decontaminated* cell of column i are either *decontaminated* or *decontaminating*; and (iv) the distance between any two consecutive entry points in column $i+1$ (for $i < n-1$) is smaller than d_{max}.

1. Base - column 0: According to the set of rules in Table 4.6, the *decontaminating* state propagates vertically in both directions on the first column. Since the maximal distance between initially consecutive decontaminating cells is d_{max} by construction, within $\lfloor \frac{d_{max}-1}{2} \rfloor$ time units all the cells on column 0 are either *decontaminating* or *decontaminated*. According to the local rules, decontamination then propagates to column 1. Since the propagation to column 1 happened for all decontaminating cells within $\lfloor \frac{d_{max}-1}{2} \rfloor$ time units from the beginning, the distance between any two consecutive entry points in column 1 cannot be greater than d_{max}. By a similar argument as for column 0, all cells in column 1 will then become *decontaminated* or *decontaminating* within other $\lfloor \frac{d_{max}-1}{2} \rfloor$ time units. Thus, within at most d_{max} time units, all the cells in column 0 and in column 1 will be either *decontaminated* or *decontaminating*. We can conclude that the base case is true.

2. Induction hypothesis: At some point during the computation, assume all cells of column i $(0 < i < n-1)$ and all their right neighbours in column $i+1$ are either in a *decontaminating* or a *decontaminated* state, their left neighbours in column $i-1$ are decontaminated, and the entry points in column $i+1$ are at a distance of at most d_{max}.

3. Induction Step: Consider column $i+1$. By induction hypothesis we know that there is a time t when all cells in column $i-1$ are decontaminated, the cells in columns i and $i+1$ are either *decontaminating* or *decontaminated* and the entry points in column $i+1$ are at maximum distance d_{max} from each other. It follows that the decontamination propagates vertically in column $i+1$ within $\lfloor \frac{d_{max}-1}{2} \rfloor$ time units from time t, thus leaving all cells in column i clean. Decontamination propagates then to column $i+2$ and within other $\lfloor \frac{d_{max}-1}{2} \rfloor$ time units all the cells in the column $i+2$ become either decontaminated or decontaminating. We can conclude that by time $t + d_{max}$ all cells in column $i+1$ together with all their right neighbours are either *decontaminated* or *decontaminating* and the cells in column i are clean, thus concluding the proof.

By the above proof, we have shown that the set of rules in Table 4.6 achieves decontamination with time immunity $t = d_{max}$. □

Placing $k > 1$ decontaminating cells roughly equidistant on the first column we obtain as a corollary:

Corollary 4.6. *With $k > 1$ decontaminating cell per time unit, temporal decontamination can be achieved when the immunity time is at least $\lceil \frac{(2n-1)}{k} \rceil$.*

We can prove the optimality of this solution, for the case of $k = 1$ when the starting decontaminating cell is a corner.

Theorem 4.7. *With a single decontaminating cell per time unit, temporal decontamination is possible if and only if the immunity time is is at least $2n - 1$.*

Proof. Decontamination is achieved by placing the initial decontaminating cell at a corner and by applying the set of rules indicated in Table 4.6. The effect of the local rules is the snake propagation of the decontaminating cell (see Figure 4.8)where a *decontaminated* cell is never in contact with an *contaminated* one for more than $(2n - 1) - 1$ time units, resulting in a situation where all nodes are simultaneously decontaminated after $n \times n$ time units.

We now prove that the immunity time cannot be $\leq 2n - 1$. We have three possible placements of the initial decontaminating cell: corner, border, and internal.

Case 1- Internal: Let us first consider the case when the initial decontaminating cell is an internal cell (i, j). In this case we will show that, regardless of the immunity time, there exists no set of rules that allows the maintenance of a single decontaminating cell. Let us call a rule that cannot be present in a correct set of rules a *forbidden rule*. Notice that $r_0 = \{\bullet, *, *, *, *, *, *, *, *\} \rightarrow 0$ is necessarily part of the set of rules to ensure the presence of a single *decontaminating* cell. To allow the spread of decontamination to a single new cell, one (and only one) of the following eight rules must be present: $r_1 = \{1, \bullet, 1, 1, 1, 1, 1, 1, 1\} \rightarrow \bullet$; $r_2 = \{1, 1, \bullet, 1, 1, 1, 1, 1, 1\} \rightarrow \bullet$; $r_3 = \{1, 1, 1, \bullet, 1, 1, 1, 1, 1\} \rightarrow \bullet$; $r_4 = \{1, 1, 1, 1, \bullet, 1, 1, 1, 1\} \rightarrow \bullet$; $r_5 = \{1, 1, 1, 1, 1, \bullet, 1, 1, 1\} \rightarrow \bullet$; $r_6 = \{1, 1, 1, 1, 1, 1, \bullet, 1, 1\} \rightarrow \bullet$; $r_7 = \{1, 1, 1, 1, 1, 1, 1, \bullet, 1\} \rightarrow \bullet$; or
$r_8 = \{1, 1, 1, 1, 1, 1, 1, 1, \bullet\} \rightarrow \bullet$. Without loss of generality, let r_3 be present and let $r_1, r_2, r_4, r_5, r_6, r_7$, and r_8 be forbidden, because any other choice will produce a symmetric behaviour. The combination of r_0 and r_3 makes the decontaminating cell propagate down until reaching the upper neighbour of the border cell. Then the following rules become forbidden because they do not preserve the existence of a single decontaminating cell: $r_9 = \{1, 1, 1, 1, 1, 0, \bullet, 1, 1\} \rightarrow \bullet$; $r_{10} = \{1, 1, 1, 1, 0, \bullet, 1, 1, 1\} \rightarrow \bullet$; $r_{11} = \{1, \bullet, 0, 1, 1, 1, 1, 1, 1\} \rightarrow \bullet$; $r_{12} = \{1, 0, 1, 1, 1, 1, 1, 1, \bullet\} \rightarrow \bullet$;

$r_{13} = \{0, 1, 1, 1, 1, 1, 1, \bullet, 1\} \to \bullet$; $r_{14} = \{1, 0, 0, 1, 1, 1, 1, 1, \bullet\} \to \bullet$; and
$r_{15} = \{1, 1, 1, 1, 0, 0, \bullet, 1, 1\} \to \bullet$).

At this point, to ensure the propagation of the decontaminating cell, one (and only one) of the following three rules must be added to the set: $r_{16} = \{1, 1, 1, \bullet, 1, 1, 0, 0, 0\} \to \bullet$; $r_{17} = \{1, 1, 1, 1, \bullet, 1, 0, 0, 0\} \to \bullet$; or $r_{18} = \{1, 1, \bullet, 1, 1, 1, 0, 0, 0\} \to \bullet$. Without loss of generality, let r_{16} be present and r_{17}, and r_{18} be forbidden, r_{17} and r_{18} will produce a symmetric behaviour. Rule r_{16} propagates the decontaminating cell to the border. The decontaminating cell is now the one in the border, one (and only one) of the following rules must be added to the set: $r_{19} = \{1, 1, 1, 1, 0, \bullet, 0, 0, 0\} \to \bullet$; $r_{20} = \{1, \bullet, 0, 1, 1, 1, 0, 0, 0\} \to \bullet$; or $r_{21} = \{0, 1, 1, 0, 1, 1, 1, \bullet, 1\} \to \bullet$. Without loss of generality, let r_{19} be present and r_{20}, and r_{21} be forbidden, another choice will produce a symmetric behaviour.

At this point, the decontaminating cell is on the lower border of the left column, and one (and only one) of the following rules must be added to the set:
$r_{22} = \{1, 1, 1, 1, 0, 0, 0, \bullet, 1\} \to \bullet$; $r_{23} = \{1, 1, 1, 1, 1, \bullet, 0, 0, 0\} \to \bullet$;
$r_{24} = \{0, \bullet, 1, 0, 1, 1, 0, 0, 0\} \to \bullet$; or $r_{25} = \{0, 1, 1, 0, 1, 1, 1, 0, \bullet\} \to \bullet$). Without loss of generality, let r_{22} be present and r_{23}, r_{24}, and r_{25} be forbidden, any other choice will produce a symmetric behaviour. The combination of r_0 and r_{22} makes the decontaminating cell propagate up, and to avoid the presence of more than one cell in a decontaminating state the following rules must be forbidden: $r_{26} = \{1, 1, 1, 1, 1, \bullet, 0, 1, 1\} \to \bullet$; $r_{27} = \{1, 1, 1, 1, \bullet, 0, 0, 0, 0\} \to \bullet$; $r_{28} = \{0, 1, 1, \bullet, 0, 0, 0, 0, 0\} \to \bullet$;
$r_{29} = \{0, \bullet, 1, 0, 1, 1, 1, 0, 0\} \to \bullet$; and $r_{30} = \{0, 0, \bullet, 0, 1, 1, 0, 0, 0\} \to \bullet$. When the decontaminating cell reaches the lower left neighbour of the initial decontaminating cell, one (and only one) of the following two rules must be added to the set: $r_{31} = \{1, 1, 1, 1, 1, 0, 0, \bullet, 1\} \to \bullet$; or $r_{32} = \{0, 1, 1, 1, 1, 1, 1, 0, \bullet\} \to \bullet$. Without loss of generality, let r_{31} be permitted and r_{32} be forbidden, r_{32} will produce a symmetric behaviour.

The decontaminating cell is now the one to the left of the initial decontaminating cell, and to propagate the decontamination one (and only one) of the following rules must be added to the set of rules: $r_{33} = \{1, 1, 1, 1, 1, 1, 0, \bullet, 1\} \to \bullet$; $r_{34} = \{1, 1, 1, 1, 1, 1, 1, 1, 0, \bullet\} \to \bullet$; $r_{35} = \{1, 1, 1, 1, \bullet, 0, 0, 1, 1\} \to \bullet$; $r_{36} = \{0, 1, 1, \bullet, 0, 0, 0, 0, 1\} \to \bullet$;
$r_{37} = \{0, 0, \bullet, 0, 1, 1, 1, 0, 0\} \to \bullet$; or $r_{38} = \{0, \bullet, 1, 1, 1, 1, 1, 0, 0\} \to \bullet$). To propagate the decontaminating cell from this point, one (and only one) of the following rules must be added to the set: $r_{39} = \{1, \bullet, 1, 1, 1, 1, 1, 0, 0\} \to \bullet$; $r_{40} = \{0, 1, 1, \bullet, 1, 0, 0, 0, 1\} \to \bullet$; or $r_{41} = \{0, 0, \bullet, 1, 1, 1, 1, 0, 0\} \to \bullet$. Without loss of generality, let r_{39} be present and r_{40}, and r_{41} be forbidden, because any other choice will produce a symmetric propagation in

the cellular automata. To propagates the decontaminating cell onward, one (and only one) of the following rules must be added to the set: $r_{42} = \{1, 1, 1, 1, 1, 1, 1, \bullet, 0\} \to \bullet$; $r_{43} = \{1, 1, 1, 1, 1, 1, \bullet, 0, 1\} \to \bullet$; or $r_{44} = \{0, 1, 1, 1, 1, \bullet, 0, 0, 1\} \to \bullet$. Without loss of generality, let r_{42} be present and r_{43}, and r_{44} be forbidden, because r_{43}, and r_{44} will produce a symmetric behaviour.

At this point, one (and only one) of the following rules must be added to the set of rule to ensure the propagation of decontamination: $r_{45} = \{1, 1, 1, 1, 1, \bullet, 0, 0, 1\} \to \bullet$; $r_{46} = \{1, \bullet, 1, 1, 1, 1, 1, 1, 0\} \to \bullet$; $r_{47} = \{0, 1, 1, 1, \bullet, 0, 0, 0, 1\} \to \bullet$;
$r_{48} = \{1, 0, \bullet, 1, 1, 1, 1, 1, 0\} \to \bullet$; and $r_{49} = \{0, 0, 1, \bullet, 1, 1, 1, 0, 0\} \to \bullet$. Without loss of generality let r_{45} be present and r_{46}, r_{47}, r_{48}, and r_{49} be forbidden, because any other choice will produce a symmetric propagation in the cellular automata. The combination of $r_0, r_{33}, r_{39}, r_{42}$, and r_{45} propagates the decontamination up to the lower neighbour of the upper border while oscillating between the column j and $j-1$. The following rules than become forbidden: $r_{50} = \{1, 1, 1, 1, 1, \bullet, 0, 1, 1\} \to \bullet$; $r_{51} = \{0, \bullet, 1, 1, 1, 1, 1, 0, 0\} \to \bullet$; $r_{52} = \{0, 1, 1, \bullet, 0, 0, 0, 0, 1\} \to \bullet$; and $r_{53} = \{0, 0, \bullet, 0, 1, 1, 1, 0, 0\} \to \bullet$. Now, to propagate the decontamination to the upper border, one (and only one) of the following rules must be added to the set: $_{54} = \{1, 1, 0, 0, 0, \bullet, \} \to \bullet$; or $r_{55} = 1, 1, 0, 0, 0, 1, 1, 0, \bullet \to \bullet$. Without loss of generality, let r_{54} be present and r_{55} be forbidden, because any other choice will produce a symmetric propagation in the cellular automata. One (and only one) of the following two rules must be present to propagate the decontamination: $r_{56} = \{1, 1, 0, 0, 0, \bullet, 0, 0, 1\} \to \bullet$; or $r_{57} = \{1, \bullet, 0, 0, 0, 1, 1, 0, 0\} \to \bullet$. Without loss of generality, let r_{57} be present and r_{56} be forbidden, since r_{56} and r_{56} will produce a symmetric propagation in the cellular automata.

Only the two columns j and $j+1$ are decontaminated, and the time for decontaminating these two columns is $2n-1$. The initial decontaminating cell has four neighbours in columns $j+1$ and four neighbours in columns $j-1$, which are still contaminated and the required immunity must be higher then $2n-1$, which contradicts our hypothesis. It follows that starting from an internal cell, we cannot decontaminate all cells with a single decontaminating cell per time unit within $2n-1$ time immunity.

Case 2- Border: Let us consider the case when the initial decontaminating cell is a border cell. Four placements are possible: $(i, 0), (i, n), (0, j)$, and (n, j). Without loss of generality, let us consider the case where the initial cell is the border cell $(i, 0)$. Notice that $r_0 = \{\bullet, *, *, *, *, *, *, *, *\} \to 0$ is necessarily part of the set of rules to ensure the presence of a single decontaminating cell. To allow the spread of decontamination to a single new cell, one (and only one) of the following five rules must be present: $r_1 =$

$\{1,0,0,\bullet,1,1,1,1,0\} \to \bullet$; $r_2 = \{1,0,0,1,1,1,1,\bullet,0\} \to \bullet$; $r_3 = \{1,1,1,1,1,1,1,1,\bullet\} \to \bullet$; $r_4 = \{1,\bullet,1,1,1,1,1,1,1\} \to \bullet$; or $r_5 = \{1,1,\bullet,1,1,1,1,1,1\} \to \bullet$. Without loss of generality, let r_1 be present and r_2, r_3, r_4, and r_5 be forbidden, any other choice will produce a symmetric behaviour. The combination of r_0 and r_1 propagates the decontamination down to the upper neighbour of the cell on the border of the first column. However, to maintain a single decontaminating cell the following rules become forbidden: $r_6 = \{0,0,0,1,1,1,1,\bullet,0\} \to \bullet$; $r_7 = \{1,0,1,1,1,1,1,1,\bullet\} \to \bullet$; $r_8 = \{1,\bullet,0,1,1,1,1,1,1\} \to \bullet$; and $r_9 = \{1,0,0,1,1,1,1,1,\bullet\} \to \bullet$. At this point, to propagate the decontamination, one (and only one) of the following two rules must be present: $r_{10} = \{1,0,0,\bullet,1,1,0,0,0\} \to \bullet$; or $r_{11} = \{1,1,\bullet,1,1,1,0,0,0\} \to \bullet$. Without loss of generality, let r_{10} be present and r_{11} be forbidden. The combination of r_0 and r_{10} propagates the decontamination to the lower left border.

At this point, $r_{12} = \{1,\bullet,0,1,1,1,0,0,0\} \to \bullet$ is the only possible rule that could be added to the to the set of rules. To ensure the propagation of the decontaminating cell, one (and only one) of the following two rules must be added to the set of rules: $r_{13} = \{1,\bullet,1,1,1,1,0,0,0\} \to \bullet$; or $r_{14} = \{0,0,0,0,1,\bullet,0,0,0\} \to \bullet$, otherwise r_{14} is forbidden. The decontaminating cell would propagate back to the original cell because of the fact that r_{12}; and r_9 are forbidden, and the only possible movement would be to the right again, giving rise to an oscillation. The combination of r_0 and r_{13} propagate the decontamination cell until reaching the left neighbour of the cell on the border, and then the following rules become forbidden because they prevent the existence of a single cell in a*decontaminating* state: $r_{15} = \{1,1,1,1,1,1,1,\bullet,0\} \to \bullet$; and $r_{16} = \{1,0,0,1,1,1,\bullet,0,0\} \to \bullet$. To propagate the decontamination onward, one (and only one) of the following three rules must be added to the set of rules: $r_{17} = \{1,\bullet,1,1,0,0,0,0,0\} \to \bullet$; $r_{18} = \{1,1,1,1,0,0,0,1,\bullet\} \to \bullet$; or
$r_{19} = \{1,1,1,1,1,1,\bullet,0,0\} \to \bullet$. Without loss of generality let r_{17} be present and r_{18}, and r_{19} be forbidden. At this point, the *decontaminating* cell is the cell at the lower right corner, and the only possible rule is $r_{20} = \{1,1,1,1,0,0,0,\bullet,0\} \to \bullet$. The combination of r_0, and r_{20} propagates the decontamination to the lower neighbour of the upper right corner.

At this point in time, the neighbours of the initial decontaminating cell are still contaminated. The time required to decontaminate them is at least $(n-2)$ (distance between last and second column), and this time already elapsed since the initial decontaminating cell is: $3n - j - 3$ (on the first column $(n-j), (j \neq n)$, $(n-1)$ on the last row, and $(n-2)$ on the last column). The required immunity must be higher then $2n-1$, which

contradicts our hypothesis. It follows that starting from a border cell, we cannot decontaminate all cells with a single decontaminating cell per time unit within $2n-1$ time immunity.

Starting from a corner would be similar. In fact, starting from the corner would be the best (as we know), but still cannot achieve $((2n-1)-1)$. □

Corollary 4.8. *With k decontaminating cell per time unit, temporal decontamination can be achieved when the immunity time is $\geq \lfloor \frac{2n-1}{k} \rfloor$. Global decontamination can be achieved in a finite cellular automata with the Moore neighbourhood and immunity time t using $\lfloor \frac{2\times n}{t} \rfloor$ simultaneous decontaminating cells.*

4.2.2 Circular Cellular Automata

In this section we will briefly discuss the case of circular cellular automata with both Von Neumann and Moore neighbourhoods.

4.2.2.1 Circular Cellular Automata with the Von Neumann Neighbourhood

A straightforward generalization of the strategy for the finite case is not possible with Von Neumann neighbourhood. In fact, we have the following negative result:

Theorem 4.9. *With $1, 2, 4$ decontaminating cells per time unit, temporal decontamination in a circular cellular automata with the Von Neumann neighbourhood is not possible regardless of the immunity time.*

Proof. Since the cells of the automata are totally symmetric, the initial *decontaminating* cell can be any cell in the cellular automata. Notice that $r_0 = \{\bullet, *, *, *, *\} \rightarrow 0$ is necessarily part of the set of rules to ensure the presence of a single *decontaminating* cell. To allow the spread of decontamination to a single new cell, one (and only one) of the following four rules must be present $r_1 = \{1, \bullet, 1, 1, 1\} \rightarrow \bullet$; $r_2 = \{1, 1, \bullet, 1, 1\} \rightarrow \bullet$; $r_3 = \{1, 1, 1, \bullet, 1\} \rightarrow \bullet$; or $r_4 = \{1, 1, 1, 1, \bullet\} \rightarrow \bullet$). Without loss of generality, let r_1 be present and let r_2, r_3, and r_4 be forbidden because any other choice will produce a symmetric propagation in the cellular automata. However, the combination of r_0 and r_1 propagate the decontamination to the right until reaching the left neighbours of the initial decontaminating cell. At this point, since r_2 and r_4 are forbidden, the only possible rules which do not stop the decontamination are: $r_5 = \{0, \bullet, 1, 0, 1\} \rightarrow \bullet$; and

$r_6 = \{0, 0, 1, \bullet, 1\} \rightarrow \bullet$, adding r_5 or r_6 to the set of rules will make the decontamination oscillate without propagation to other contaminated cells. □

Furthermore, we conjecture that

Conjecture 4.10. *With $k < n$* decontaminating *cells per time unit, temporal decontamination in a circular cellular automata with the Von Neumann neighbourhood is not possible regardless of the immunity time.*

4.2.2.2 Circular Cellular Automata with the Moore Neighbourhood

With the Moore neighbourhood we can easily generalize the strategy employed for the finite case to obtain decontamination.

We place initial *decontaminating* cells equidistant on the first column at the maximum possible distance t (t = immunity time). We design the rules in such a way that the decontamination propagates in both directions on the first column until two *decontaminating* cells become adjacent (or a *decontaminating* cell has the two neighbours on the column decontaminated). At this point, the rules will let the decontamination propagate in both directions (right and left) to the second column and last column. The procedure continues until two decontaminated column become adjacent or a column has both left and right column neighbours decontaminated (see Figure 4.10). The rules are described in Table 4.7 and it is easy to see that this strategy extends the one described for the finite case.

Theorem 4.11. *Let d_{max} be the maximum distance between two consecutive* decontaminating *cells or between a* decontaminating *cell and a corner at time 0. If the initial number of* decontaminating *cells is at least 2, the set of rules given in Table 4.7 achieves decontamination in a circular cellular automata with immunity time $t = d_{max}$.*

Configuration	Next State
$\{\bullet, *, *, *, *, *, *, *, *\}$	0
$\{1, 1, 1, \bullet, 1, 1, 1, 1, 1\}$	$\bullet$
$\{1, 1, 1, 1, 1, 1, 1, \bullet, 1\}$	$\bullet$
$\{1, 1, 1, \bullet, 1, 1, 1, \bullet, 1\}$	$\bullet$
$\{1, \bullet, 0, 1, 1, 1, 1, 1, \bullet\}$	$\bullet$
$\{1, \bullet, 0, 1, 1, 1, 1, 1, 0\}$	$\bullet$
$\{1, 1, 1, 1, 0, \bullet, 0, 1, 1\}$	$\bullet$
$\{1, 1, 1, 1, \bullet, \bullet, 0, 1, 1\}$	$\bullet$
$\{1, 0, 0, \bullet, 1, 1, 1, 1, 0\}$	$\bullet$
$\{1, 0, 0, 1, 1, 1, 1, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 1, 1, 1, \bullet, 0\}$	$\bullet$
$\{1, 1, 1, 1, 0, 0, 0, \bullet, 1\}$	$\bullet$
$\{1, 1, 1, \bullet, 0, 0, 0, 1, 1\}$	$\bullet$
$\{1, 1, 1, \bullet, 0, 0, 0, \bullet, 1\}$	$\bullet$
$\{1, \bullet, 0, 1, 0, 0, 0, 1, 0\}$	$\bullet$
$\{1, 0, 0, 1, \bullet, \bullet, 0, 1, \bullet\}$	$\bullet$
$\{1, 0, 0, 1, 0, \bullet, 0, 1, \bullet\}$	$\bullet$
$\{1, 0, 0, \bullet, 1, \bullet, 0, 1, 0\}$	$\bullet$
$\{1, 0, 0, 1, 0, \bullet, 1, \bullet, 0\}$	$\bullet$
$\{1, \bullet, 1, \bullet, 0, 0, 0, 1, 0\}$	$\bullet$
$\{1, 0, 0, 1, \bullet, \bullet, 0, 1, \bullet\}$	$\bullet$
$\{1, \bullet, 0, 1, 0, 0, 0, 1, 1\}$	$\bullet$
$\{1, 0, 0, 1, \bullet, 1, 1, \bullet, 0\}$	$\bullet$
$\{1, 1, \bullet, 1, 0, 0, 0, \bullet, 1\}$	$\bullet$
$\{1, \bullet, 0, 1, 0, \bullet, 0, 1, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 0, 0, 0, 1, 0\}$	$\bullet$
$\{1, 0, 0, 1, 0, 0, 0, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 0, 0, 0, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, 0, 0, 0, 0, \bullet, 0\}$	$\bullet$
$\{1, 0, 0, \bullet, 0, 0, 0, 0, 0\}$	$\bullet$

Table 4.7: Rule Table for Temporal Decontamination with Multiple Decontaminating cells in Circular Cellular Automata with the Moore Neighbourhood.

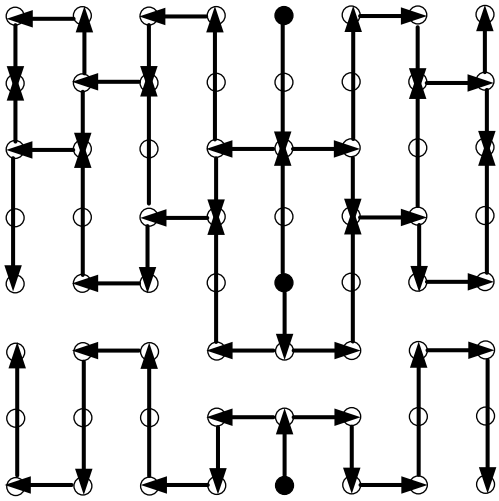

Figure 4.10: Propagation of Multiple Decontaminating cells with Temporal Immunity and the Moore Neighbourhood

4.3 Conclusion

In this chapter, we have considered the problem of decontaminating two-dimensional three-state cellular automata with and without temporal immunity. We have focused on finite and circular cellular automata with Von Neumann and Moore neighbourhoods. In each case we have described cellular automata rules to achieve decontamination. We have measured the efficiency of our sets of rules by considering the number of simultaneous cells in a *decontaminating* state and, with this metric, we have shown that most rules are optimal. We have described rules achieving optimal decontamination for finite cellular automata with the Von Neumann neighbourhood which employ single, two, and four simultaneously decontaminating cells, but no optimal strategy has been proposed for three simultaneously decontaminating cells or for any number between 4 and $n-1$. For the Moore neighbourhood a set of rules is given for k, $k \geq 1$ decontaminating cells. In the case of circular cellular automata, we proved that decontamination with single, two, and four simultaneously decontaminating cells is not possible with the Von Neumann neighbourhood and we provide a set of rules in the case of the Moore neighbourhood. Table 4.8 summarizes the results obtained in this chapter.

Neighbourhood	Number of Decontaminating Cells	Immunity Time
Finite Cellular Automata		
Von Neumann	$k = n$	$t = 1^*$
	$k = 1, 2, 4$	$t \geq \frac{4}{k}(n-1) - 1^*$
Moore	k	$t \geq \lceil \frac{(2n-1)}{k} \rceil$
	$k = 1$	$t \geq (2n-1)^*$
Circular Cellular Automata		
Von Neumann	$k = 2n$	$t = 1^*$
Moore	$k \geq 2$	$t = {d_{max}}^\dagger$

Table 4.8: Chapter 4: Summary of Results.

*The proposed strategy is an optimal solution

†d_{max} the maximum distance between two consecutive *decontaminating* cells or between a *decontaminating* cell and a corner at time 0.

Chapter 5

Decontamination by Mobile Cellular Automata

In this chapter we focus on the decontamination problem by mobile cellular automata. We consider both basic and temporal decontamination, and we look at both Von Neumann and Moore neighbourhoods. Remember that in the case of mobile cellular automata, the active elements (i.e., the agent) are *oblivious* (i.e., they do not remember their past actions) and the decision regarding their movement at some time t is solely based on their current neighbourhood. In these settings, we will consider two sets of solutions depending on whether or not the agents have *cloning* capabilities, and we will observe what the impact of such a capability is on the efficiency of the solutions. While basic decontamination with a Von Neumann neighbourhood is quite simple and the possibility of cloning does not really have an impact on the solution, with temporal decontamination we can observe that cloning does indeed makes a difference.

Given a mobile cellular automata (finite or circular) of size $n \times n$ and an immunity time t, the goal is to design the local rules for the agents and their initial placement so that the agents can decontaminate the entire system without allowing any cell to be recontaminated. To be efficient, the decontamination should employ as few agents as possible. We design several strategies depending on the type of neighbourhood, and on the ability of the agents to clone themselves.

In [27] results obtained in this chapter for: temporal decontamination with multiple agents and without cloning in the case of Von Neumann and Moore neighbourhoods, and temporal decontamination with cloning with multiple agents in the case of Von Neumann neighbourhood, are published.

5.1 Basic Decontamination

In this section we consider basic decontamination without cloning in finite and circular two-dimensional mobile cellular automata. The scenarios considered the use of agents where the decontamination process does not provide any type of immunity, where agents do not have cloning capability, and where a cell is contaminated as soon as one of its neighbours is contaminated. In this case, any decontamination algorithm has to forbid a decontaminated cell to ever be in contact with a contaminated or potentially contaminated cell. We provide simple solutions for finite and circular mobile cellular automata.

5.1.1 Finite Mobile Cellular Automata

Notice that with basic decontamination, a single agent would obviously not be sufficient to perform decontamination. Immediately after decontaminating a cell, it would inevitably be exposed to a contaminated cell as at most one of its neighbours could become *decontaminating*. The question is what is the minimum number of agents which could guarantee decontamination without recontamination. We prove in the following that n agents are necessary and sufficient.

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, \bullet, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, 0\}$	0	*Terminate*

Table 5.1: Rule Table for Basic Decontamination in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood

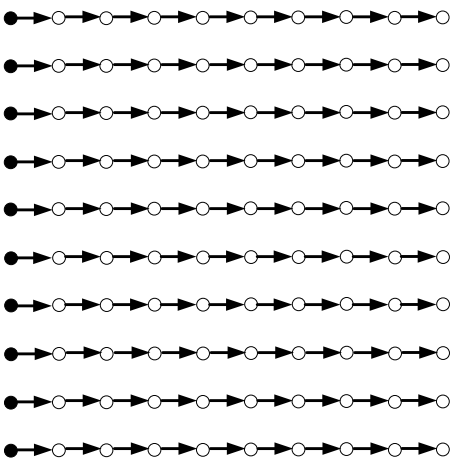

Figure 5.1: Basic Decontamination in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood

Theorem 5.1. *Optimal basic decontamination can be achieved in a finite $n \times n$ mobile cellular automata with a Von Neumann neighbourhood using n agents.*

Proof. To prove that n agents are sufficient, we need to prove that starting with n agents initially located at the first column, all cells will become *decontaminated* by the end of the process, and no cell will be recontaminated. That is, once *decontaminated*, every cells stays *decontaminated* until the end of the process, when all cells are *decontaminated*. Starting with n agents initially located at the first column, obviously our algorithm visit all columns synchronously moving right to the next column at each time step.

To prove optimality, by contradiction let us assume that $n - 1$ agents are sufficient to achieve decontamination in a finite mobile cellular automata.

Notice that if a row or a column is without an agent and with a contaminated cell, then it must be completely contaminated (if the neighbouring cell on that row or column is decontaminated then it will become contaminated in the next time step, and this will contradict the monotonicity of the decontamination process).

Consider the first time during the decontamination process that an agent reaches an arbitrary cell c in the last row, i.e. c is now guarded. At most $(n-1)$ agents are available and the mobile cellular automata is composed of n rows, thus there must be at least an empty row r_i where $(0 \leq i < n - 1)$. This row has to be *decontaminated*, otherwise, if one node is contaminated, then the whole row r_i is contaminated as well (according to the above observation). This, however, contradicts the monotone decontamination

assumption. In fact, r_i would divide the mobile cellular automata into two disconnected regions, one containing at least $n-1$ *decontaminated* (or *decontaminating*) cells (the initial placement of agents), and the other one containing a *decontaminated* cell (c) and at least one contaminated cell (any cell in the last row different from cell c). Thus, row r_i must be *decontaminated*. As the number of available agents is $(n-1)$, there must be at least an empty column c_j where $(0 < j < n)$. Column c_j clearly cannot contain c, c_j intersect the last row the last row in a contaminated cell b, otherwise b was decontaminated with an agent from column c_j, contradicting that c_j has no agent. However, at this instant of time c_j is contaminated as the node in its last row is contaminated (according to the above observation). The entire column c_j is contaminated but c_j intersects r_i, and so r_i cannot be a *decontaminated* row. This contradiction shows that, the decontamination in a $n \times n$ mobile cellular automata with a Von Neumann neighbourhood requires at least n agents. □

5.1.2 Circular Mobile Cellular Automata

We propose a solution where initially all agents are located in two adjacent columns in circular mobile cellular automata, agents move in opposite direction to decontaminate the whole circular mobile cellular automata. We prove in the following that $2n$ agents are necessary and sufficient.

Configuration	Next State	Agents Movement
$\{\bullet, \bullet, \bullet, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, \bullet, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 1, \bullet, \bullet, \bullet\}$	0	$\Leftarrow$
$\{\bullet, 1, \bullet, 0, \bullet\}$	0	$\Leftarrow$
$\{\bullet, 0, \bullet, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, \bullet\}$	0	*Terminate*

Table 5.2: Rule Table for Basic Decontamination in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood

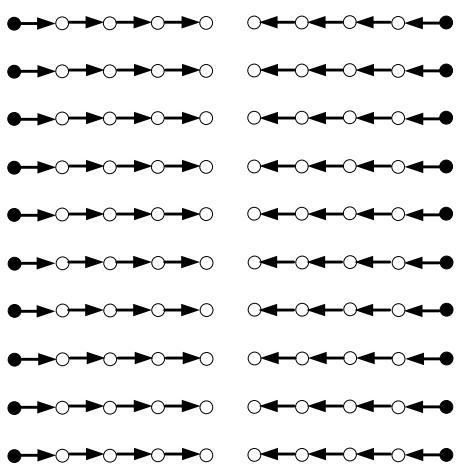

Figure 5.2: Rule Table for Basic decontamination in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood

Theorem 5.2. *Optimal basic decontamination can be achieved in a circular mobile cellular automata with a Von Neumann neighbourhood using* $2n$ *agents.*

Proof. To prove that $2n$ agents are sufficient, we provide a strategy that uses $2n$ agents (see Table 5.2). We initially place agents in the cells of any two adjacent columns, the agents placed in the right column move to the left and the agents placed in the left column move to the right. An agent terminates when it sees that the cells in the direction it is supposed to move are either *decontaminating* or *decontaminated* (see Figure 5.2).

Following the same lines as the arguments of Theorem 4.2 we also obtain a lower bound for the circular mobile cellular automata. The minimum perimeter occurring if the shape of the block were to be a rhombus would be $4\sqrt{f}-4$ and the area $f = l^2+(l-1)^2$, thus the number of required agents would be $X \geq 2\sqrt{2f-1}-2$. Consider a moment during the cleaning when there are $f = l^2 + (l-1)^2 \geq \frac{n^2+2n+2}{2}$ decontaminated cells. This holds as long as $n \times n = n^2 \geq \frac{n^2+2n+2}{2}$ which gives $n > 2$. From $f \geq \frac{n^2+2n+2}{2}$, it follows that the number of required agents is $X \geq 2\sqrt{2(\frac{n^2+2n+2}{2})-1}-2 = 2n$. □

5.2 Basic Decontamination with Cloning

In this section we consider basic decontamination with cloning in finite and circular two-dimensional three state mobile cellular automata.

5.2.1 Finite Mobile Cellular Automata: Optimal Strategy

Decontamination is achieved by placing a single agent at a corner and by applying the set of rules indicated in Table 5.3. Figure 5.3 show the propagation of decontamination starting with a single agent located at the left most corner.

Starting with a single agent located in one of the four corners of the mobile cellular automata, an agent clones itself whenever it has more than one neighbour. If two agents meet in the same cell, they fuse. The number of agents increases until the agents move to the central diagonal which requires n agents to decontaminate the cells on the central diagonal and avoid re-contamination. The number of agents will then start decreasing until reaching the opposite corner until becoming a single agent.

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow\Rightarrow$
$\{\bullet, 0, \bullet, 1, \bullet\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 1, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, 0\}$	0	*Terminate*

Table 5.3: Rule Table for Basic Decontamination with Cloning in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood

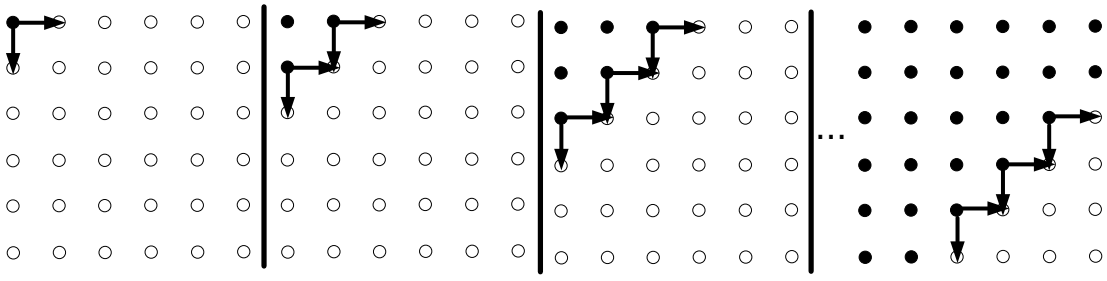

Figure 5.3: Basic Decontamination with Cloning in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood

Theorem 5.3. *Optimal basic decontamination with cloning can be achieved in a finite $n \times n$ mobile cellular automata with a Von Neumann neighbourhood using n agents.*

Proof. To prove that n agents are sufficient, we show that the set of rules given in Table 5.3 achieves the decontamination. Starting with a single agent initially located at corner, all cells will become decontaminated by the end of the process, the number of agent never become greater than n, and no cell will be re-contaminated. As described in Table 5.3, each time an agent "see" that there is two contaminated cells it will clone itself and move on that cells. Traversing the finite mobile cellular automata orderly from diagonal 1 to diagonal $(2n-1)$ (see Figure 5.3), the largest diagonal will need n agents. Obviously this proves the sufficiency.

Following the same lines as the arguments of Theorem 5.1 we also obtain a lower bound (n agents) also for the finite mobile cellular automata decontamination with cloning. □

5.2.2 Circular Mobile Cellular Automata: Optimal Strategy

Starting with four agents located at the corners (Figure 5.4) the number of agents increases until reaching $2n$. Decontamination is achieved by placing four agents at the corner and by applying the set of rules indicated in Table 5.4. Each agent moves and clones itself whenever it has a contaminated neighbour. If two agents meet in the same cell they will fuse. An agent terminates when it cannot see more contaminated cells.

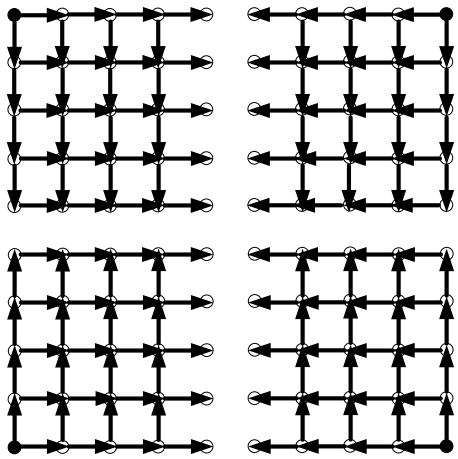

Figure 5.4: Basic Decontamination with Cloning in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood

Configuration	Next State	Agents Movement
$\{\bullet, \bullet, \bullet, 1, 1\}$	0	$\Downarrow\Rightarrow$
$\{\bullet, 1, \bullet, \bullet, 1\}$	0	$\Downarrow\Leftarrow$
$\{\bullet, 1, 1, \bullet, \bullet\}$	0	$\Uparrow\Leftarrow$
$\{\bullet, \bullet, 1, 1, \bullet\}$	0	$\Uparrow\Rightarrow$
$\{\bullet, 0, \bullet, 1, 1\}$	0	$\Downarrow\Rightarrow$
$\{\bullet, \bullet, 0, 1, 1\}$	0	$\Downarrow\Rightarrow$
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow\Rightarrow$
$\{\bullet, \bullet, 1, 1, 0\}$	0	$\Uparrow\Rightarrow$
$\{\bullet, 0, 1, 1, \bullet\}$	0	$\Uparrow\Rightarrow$
$\{\bullet, 0, 1, 1, 0\}$	0	$\Uparrow\Rightarrow$
$\{\bullet, 1, \bullet, 0, 1\}$	0	$\Downarrow\Leftarrow$
$\{\bullet, 1, 0, \bullet, 1\}$	0	$\Downarrow\Leftarrow$
$\{\bullet, 1, 0, 0, 1\}$	0	$\Downarrow\Leftarrow$
$\{\bullet, 1, 1, 0, \bullet\}$	0	$\Uparrow\Leftarrow$
$\{\bullet, 1, 1, \bullet, 0\}$	0	$\Uparrow\Leftarrow$
$\{\bullet, 1, 1, 0, 0\}$	0	$\Uparrow\Leftarrow$
$\{\bullet, 0, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 1, \bullet, 0, 0\}$	0	$\Leftarrow$
$\{\bullet, 1, 0, 0, \bullet\}$	0	$\Leftarrow$
$\{\bullet, 0, \bullet, \bullet, 1\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, 0, 1\}$	0	*Terminate*
$\{\bullet, 0, 0, \bullet, 1\}$	0	*Terminate*
$\{\bullet, \bullet, 0, 0, 1\}$	0	*Terminate*
$\{\bullet, 0, 1, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, \bullet, 1, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, 1, \bullet, 0\}$	0	*Terminate*
$\{\bullet, \bullet, 1, 0, 0\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, 1, 0\}$	0	*Terminate*
$\{\bullet, \bullet, 0, 1, \bullet\}$	0	*Terminate*
$\{\bullet, 0, 0, 1, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 1, 0\}$	0	*Terminate*
$\{\bullet, 1, \bullet, \bullet, 0\}$	0	*Terminate*
$\{\bullet, 1, \bullet, 0, 0\}$	0	*Terminate*
$\{\bullet, 1, 0, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, 1, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, \bullet, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, 0, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, \bullet, 0\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, 1\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, 1\}$	0	*Terminate*
$\{\bullet, \bullet, 0, 1, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 1, 0\}$	0	*Terminate*
$\{\bullet, 1, 0, \bullet, 0\}$	0	*Terminate*
$\{\bullet, 1, 0, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 1, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, 1, 0, 0\}$	0	*Terminate*
$\{\bullet, \bullet, 0, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, \bullet, 0\}$	0	*Terminate*

Table 5.4: Rule Table for Basic Decontamination with Cloning in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood

Theorem 5.4. *Optimal basic decontamination with cloning can be achieved in a circular $n \times n$ mobile cellular automata with a Von Neumann neighbourhood using $2n$ agents.*

Proof. We prove that $2n$ agents are sufficient by showing that: the set of rules given in Table 5.4 achieves decontamination, all cells will become decontaminated by the end of the process; the number of agents never become greater than $2n$; and no cell will be recontaminated. That is, once decontaminated every cell stays decontaminated until the end of the process, when all cells are decontaminated. Initially agents are located at cells $(0,0)$, $(0,n-1)$, $(n-1,0)$, and $(n-1,n-1)$, agents move on the diagonals and each time an agent see 2 contaminated cells it will clone itself since the agent move synchronically they will sees other agents when it reach the middle of the first columnar this point the number of agents in each part will be $\frac{n}{2}$, and the total number of agents is $2n$. Since whenever an agent "sees" a more than one contaminated cell it will clone itself so once decontaminated the contaminated neighbours of cell will be decontaminated at the next time step, thus no recontamination can occurs.

Following the same lines as the arguments of Theorem 5.2 we also obtain a lower bound ($2n$ agents) also for the circular mobile cellular automata decontamination with cloning. □

5.3 Temporal Decontamination

The case of temporal decontamination is quite different from basic decontamination. With temporal decontamination we must design a set of local rules and choose homebases from which agents start the decontamination in such a way that during the decontamination, a *decontaminated* cell never comes into contact with a *contaminated* cell after its immunity time has expired.

As mentioned earlier, without being able to generate agents (cloning) if we want to maintain a constant number of agents in the system we must design rules that force the agents to not move into the same cell. In order to achieve this we need to either increase their neighbourhood (i.e., visibility radius) or prescribe a very specific initial configuration. We will look at both cases.

5.3.1 Finite Mobile Cellular Automata with a Von Neumann Neighbourhood

In the following we describe two sets of rules (Tables 5.5 and 5.6). Both require the agents to start in a particular configuration in the first column. Depending on time immunity t, and the size n of the cellular automata we choose the strategy that uses the smaller number of agents.

5.3.1.1 Multiple Agents

We first examine the case when one agent is at the top-left corner, one at the bottom-left corner and the other agents are in groups of two at some odd distance from each other. In other words, we divide n in equal groups of $d+1$ consecutive cells and employ 2 agents in each group, one on top, and one on the bottom (let us call such a pair *sibling agents*). In this way, the two siblings are separated by an even number $(d-1)$ of cells (see Figure 5.5). The set of rules is given in Table 5.5 and it corresponds to the movement of each group of siblings in opposite directions in the column, turning right as soon as they become adjacent. Note that adjacent agents in a column can recognize themselves as siblings (and thus turn right) because their other neighbour in the same column's *decontaminated*. On the other hand, two adjacent agents in a column are not siblings when their other neighbours are contaminated (in which case they move in opposite directions). Also note that this technique is possible only because the two siblings become adjacent but never move on the same cell, hence the need for placing them at an odd distance (i.e., separating them with an even number of cells).

Strategy 1: siblings at an odd distance. This strategy applies when the distance between two sibling is odd.

Initial placement. We place one agent at the top-left corner, one at the bottom-left corner and the other agents in groups of two at distance t_1 from each other. In other words, we divide n in equal groups of t_1+1 consecutive cells and employ 2 agents in each group, one on top, and one on the bottom (*sibling agents*). In this way the two siblings are separated by an even number (t_1-1) of cells (see Figure 5.5).

Pattern of Movement. The set of rules is given in Table 5.5 where all missing combinations of states for the neighbourhood of the cell (i,j) leave $x_{i,j}$ unchanged. These rules correspond to the movement of each group of siblings in opposite directions in the column, turning right as soon as they become adjacent. Note that adjacent agents in a

column can recognize whether they have to turn right (because their other neighbours in the same column are *decontaminated*) or whether they have to move in the column in opposite directions (because their neighbours in the same column are *contaminated*). Note that this technique is possible only because the two siblings become adjacent but never move on the same cell, hence the need for placing them at an odd distance (i.e., separating them with an even number of cells).

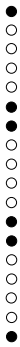

Figure 5.5: Initial Configuration: Sibling at an Even Distance

Figure 5.6: Initial Configuration: Sibling at an Odd Distance

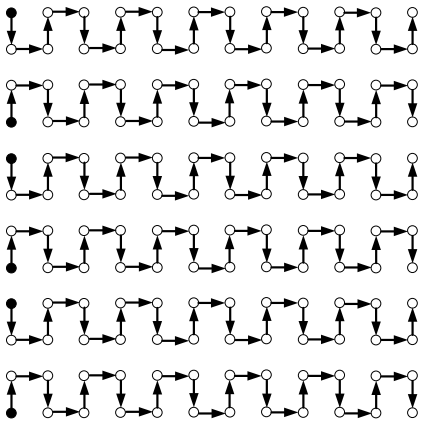

Figure 5.7: Temporal Decontamination by Multiple Agents in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood: Strategy 1.

Strategy 1 will obviously lead to the following theorem which we will omit the proof.

Theorem 5.5. *Let t_1 be the largest integer smaller than or equal to t such that $t_1 + 1$ divides n and let t_1 be odd. Temporal decontamination can be achieved in a mobile cellular*

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, 0\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, 0\}$	0	*Terminate*

Table 5.5: Rule Table for Temporal Decontamination by Multiple Agents in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood: Strategy 1.

automata with a Von Neumann neighbourhood and an immunity time t using $k = \frac{2n}{(t_1+1)}$ agents.

Strategy 2: siblings at even distance. This strategy is applied when the distance between two sibling is even.

In the following we describe a technique that would also work correctly when the distance between two siblings is even (i.e., they are separated by an odd number of cells) (see Figure 5.6). The idea is simple; it consists of employing a third agent in each interval to be placed in the central node between any two siblings. The third agent's (called the *delimiter*) only role is to keep a separation between the two intervals. The siblings move like before, but they turn to the right to move to the next column when they come in contact with the delimiter agent (see Table 5.6).

Initial placement. The idea is to place two siblings at an even distance t_1 (i.e., separated by $t_1 - 1$ cells) and employ a third agent in each interval between siblings to be placed in the central cell. The only role of the third agent (*delimiter*) is to keep a separation between the two intervals.

Pattern of Movement. The siblings move like before, but they turn to the right to move to the next column when they come in contact with the delimiter agent (see Figure 5.8). The delimiter, on the other hand, moves to the next column when it sees its two neighbours in the same column occupied by one agent each. (The set of rules corresponding to this case is given in Table 5.6). All missing combinations of states for the neighbourhood of cell (i, j) leave $x_{i,j}$ unchanged.

Note that the delimiter agent only moves to synchronize the two adjacent cleaners. The transition function that corresponds to waiting (not indicated in the Table because it does not change its state) is: $f(\bullet, 0, 1, 1, 1) = (\bullet, \{\})$, given an immunity time t. If n does not divide $t+1$, but it divides $t+2$, we can also show the configuration for which the agent have to wait: $(\{\bullet, 0, 1, 1, 1\}, \bullet$, *Stop*). The agents could be deployed as described above. As an easy consequence we have the following theorem.

Theorem 5.6. *Let t_2 be the largest integer smaller than or equal to t such that $t_2 + 2$ divides n and let t_2 be even. Temporal decontamination can be achieved in a finite mobile cellular automata with a Von Neumann neighbourhood and an immunity time t using k agents where $k = \frac{3n}{(t_2+2)}$.*

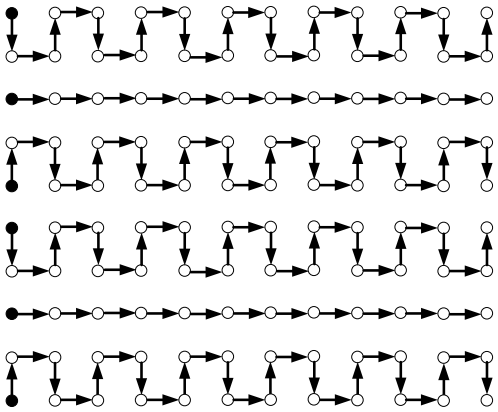

Figure 5.8: Temporal Decontamination by Multiple Agents in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood: Strategy 2.

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, \bullet, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 1, 1, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, \bullet, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, 0\}$	0	*Terminate*

Table 5.6: Rule Table for Temporal Decontamination by Multiple Agents in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood: Strategy 2.

.

5.3.1.2 Single Agent

Although the set of rules for a single agent is included in the set of rules for multiple agents, we will describe a simplified version in this section and prove correctness and optimality. In fact, for the general case with multiple agents described in the previous section, we cannot prove the optimality of either strategies. We also cannot determine, given a certain number of agents $k > 1$, the optimal immunity time t. On the other hand, when there is a single available agent, we can show that the immunity time must be at least $(2n - 1)$. This is the same result that we have obtained for the mobile agent model, and its derivation follows a very similar reasoning. We can notice that with a single agent, the capability given by *visibility* in mobile cellular automata replaces the availability of *memory* in mobile agents. In the first model the agents *can count* and

know where to move, while in the second model agents *can see* and decide based on the given set of rules where to move.

Initially the single agent is located at the left upper most corner. The rules given in Table 5.7 correspond to the movement of the agent along the first column decontaminating all cells, then the horizontal movement to the next column and so on (see Figure 5.9).

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 1, 1, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 0, 0\}$	0	*Terminate*

Table 5.7: Rule Table for Temporal Decontamination by a Single Agent in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood

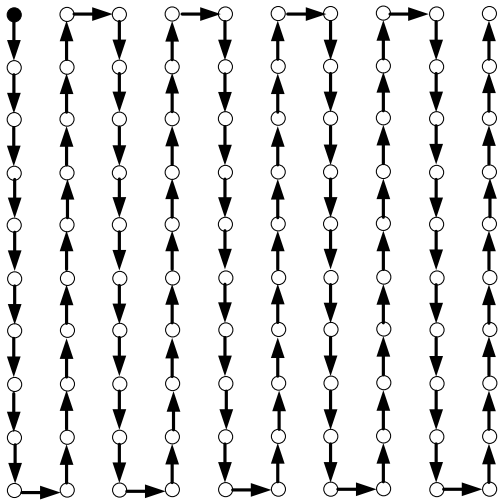

Figure 5.9: Temporal Decontamination by a Single Agent in a Finite Cellular Automata with the Von Neumann Neighbourhood

Theorem 5.7. *Optimal temporal decontamination can be achieved in a finite mobile cellular automata with a Von Neumann neighbourhood using single agent if, and only if, time immunity* $t \geq 2n - 1$.

Proof. Independent from the strategy adopted to decontaminate the finite mobile cellular automata, all the cells must be *decontaminated*. Without loss of generality, let us consider that the cell $(n-1, 0)$ at some step of the strategy $(n-1, 0)$ must be *decontaminated*.

Claim 5.8. *If the path taken by the agent a is not the vertical one, then a should visit all columns before* $(n-1, 0)$. *Therefore, it should visit all cells before* $(n-1, 0)$.

Proof. Consider $d(A, B)$ the minimum distance between A and B in the finite mobile cellular automata, P_a the path taken by the agent a, and $P_a(M, N)$ the path between M and N. If $|P_a(M, N)| = k \geq n$ then all neighbours of N must be *decontaminated* in the subgraph $P_a((0, 0), N)$. Consider that $P'_a = P_a(M, N)$ does not contain cells from column $(i + 1)$. Taking the smallest such i and the largest j in the column visited by P'_a, if the agent goes to cell $(n-1, 0)$ to decontaminate it, then the distance traversed by a since it left the first column is $(d((i-k, 0), (i, j) + d(j, i), (n-1, 0)))$ where $(k \leq i)$, which is clearly greater then n. To go back to the node $(0, j + 1)$, the same number of step is required, and if the immunity time is $2(n-1)+1$, then the cell $(0, n-1)$ will be recontaminated. The only way to ensure decontamination is to visit all the cells before the agent reach $(n-1, 0)$. □

Claim 5.9. *The Agent a must complete the decontamination of all columns* $C_a - (n-1, 0)$ *before decontaminating any cell in column* $(n-1)$.

Proof. Suppose C_a is not completely decontaminated and still a cell $(i, n-1)$ remains contaminated. Consider the largest j, such that $(j, 0)$ is *decontaminated* after the node $(i, n-1)$. Going back and forth from $(j + 1, 0)$ and $(i, n-1)$ will take more than the immunity time $2(n-1)+1$. Therefore the cell $(j, 0)$ will be contaminated again before its neighbour $(j + 1, 0)$ is *decontaminated*. □

We supposed that all cells must be visited before $(n-1, 0)$. Moreover the cell $(n-2, 0)$ is visited before any cell $(i, n-1)$ for every i, thus the path $P(n-2, 0) \to P(i, n-1) \to P(n-1, 0)$. □

5.3.2 Finite Mobile Cellular Automata with a Moore Neighbourhood

In the case of a Moore neighbourhood, we can exploit the enlarged neighbourhood visibility of the agents to design a strategy that decontaminates the network starting from a less restricted initial placement of the agents.

The idea is to divide the finite mobile cellular automata into a set of blocks of almost equal size. If $t+1$ divides $2n$, then we create $\frac{2n}{t+1}$ blocks. In each block we place an agent, the movement of the agents will alternate "bottom to top" and "top to bottom" between consecutive blocks. In case where $t+1$ does not divides $2n$, that is $2n = r \times (t+1) + q(q > 0)$, then we create r equal blocks of size $\lceil \frac{t+1}{2} \rceil$, then we place a block of size 1 (row), with a single moving horizontally, then a block of size $q-1$. The movement of the agent on the block of size $(q-1)$ will be the opposite of the previous of size r. The block of size 1 will be used to synchronize the movements of the agent on the last block of size r and the block of size $q-1$.

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 0, 0, 1, 1, 1, 0\}$	0	$\Downarrow$
$\{\bullet, 0, 0, \bullet, 1, 1, 1, 1, 0\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 1, 1, 1, 1, 0\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 0, 0, 1, 0\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 0, 0, 0, 1, 0\}$	0	$\Downarrow$
$\{\bullet, 0, 0, \bullet, 1, 1, 1, 0, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 0, 1, 1, 1, \bullet, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, \bullet, 1, 1, \bullet, 0, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 1, 1, 1, 1, \bullet, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 1, 1, 0, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 1, 1, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 0, 0, 0, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 0, 0, 0, \bullet, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 0, 0, 0, 0, \bullet, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, \bullet, 0, 0, 0, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, \bullet, 0, 0, 0, \bullet, 0\}$	0	*Terminate*

Table 5.8: Rule Table for Temporal Decontamination by Multiple Agents in a Finite Mobile Cellular Automata with the Moore Neighbourhood

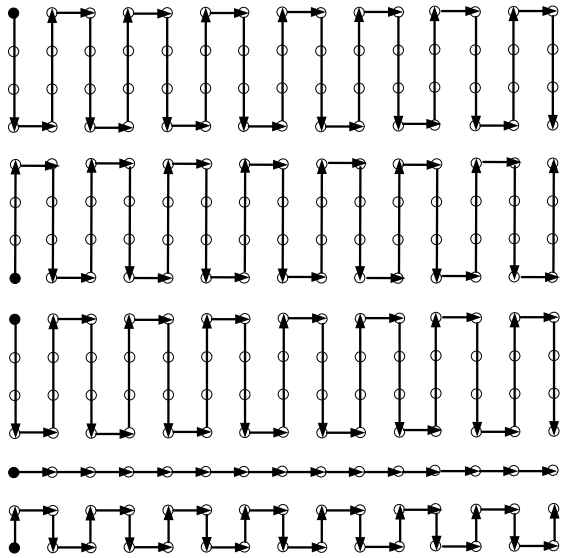

Figure 5.10: Temporal Decontamination by Multiple Agents in a Finite Mobile Cellular Automata with the Moore neighbourhood

Theorem 5.10. *Temporal decontamination can be achieved in a finite mobile cellular automata with a Moore neighbourhood and an immunity time t (where $t \geq 1$) using k agents where:*

$$k = \begin{cases} \frac{2n}{t+1} & \textit{if} \quad 2n \mod t+1 = 0 \\ \lfloor \frac{2n}{t+1} \rfloor + 2 & \textit{Otherwise} \end{cases}$$

5.3.3 Circular Mobile Cellular Automata with a Von Neumann Neighbourhood

In this section we will present a set of strategies that applies in several cases of circular mobile cellular automata with a Von Neumann neighbourhood. Most of these strategies uses the same ideas with little modifications then the once shown in the case of finite mobile cellular automata. There will be different strategies, for the cases of 1, 2, 3, and more then three agents.

A part from the single agent case where we have an optimal strategy, all other proofs are omitted since they are easy consequences from the definitions of the difference strategies.

5.3.3.1 Single Agent

With a single agent, optimal temporal decontamination in a circular mobile cellular automata is possible with time immunity $= 4(n-1)-1$ (see Table 5.9). Agent moves in spiral (see Figure 5.11).

Theorem 5.11. *With a single agent, temporal decontamination in a circular mobile cellular automata is possible if, and only if, the immunity time is* $\geq 4(n-1)-1$.

Proof. Decontamination is achieved by placing the agent in any cell and by applying the set of rules indicated in Table 5.9. The effect of the local rules is the spiral propagation of the decontamination (Figure 5.11) where a *decontaminated* cell is never in contact with a *contaminated* cell for more than $4(n-1)-1$ time units. This results in a situation where all cells are simultaneously decontaminated after $n \times n$ time units.

There is no other set of rules that achieves decontamination with a time immunity $\leq 4(n-1)-1$. Let us call a rule that cannot be present in a correct set of rules a *forbidden rule*. We now construct all possible dynamics by forbidding impossible behaviour. Initially the agent is located at any cell of the circular mobile cellular automata. Possible movements are *Left, Up, Right, Down* ($\Leftarrow$,$\Uparrow$, $\Rightarrow$, $\Downarrow$). Decontamination starts at cell x.

Without loss of generality, we can choose to propagate the decontamination in any direction because any other choice will produce a symmetric propagation in the circular mobile cellular automata. Let the decontamination propagate *Down*, $f(\bullet,1,1,1,1) = (0,\Downarrow)$ will be present and let $f(\bullet,1,1,1,1) = (0,\Leftarrow)$, $f(\bullet,1,1,1,1) = (0,\Rightarrow)$, $f(\bullet,1,1,1,1) = (0,\Uparrow)$ be forbidden. After moving to x upper neighbours, three possible moves are *Down*, *Right*, and *Left* (*Down* is forbidden because it creates an oscillation, whereas *Right* and *Left* produce a symmetric propagation.

Without loss of generality, let the agent move to the *Right* (the other choices would create a symmetric situation). At this point the only permitted rules are $f(\bullet,1,1,1,1) = (0,\Downarrow)$ and $f(\bullet,1,0,1,1) = (0,\Rightarrow)$. The agent is situated in the column to the right of the column where x is located. At this point, three possible moves are *Right*, *Up* and *Down*. The *Left* move is not possible because it leads to an oscillation. The *Up* and *Down* moves are symmetric and required to decontaminate all the columns located to the right of x to reach the left neighbour of x, and clearly require more than $4(n-1)-1$ time units. The *Right* move will propagate the decontamination to the column situated to the left of x. At this point, the *Up* and *Down* moves are permitted and both have a symmetric behaviour. Without loss of generality let us consider the *Up* propagation. The agent will decontaminate the whole column to the left of x. Then the agent will move to the left

until reaching the column to the right of x. This requires $3(n-1)+(n-2)$ moves, but the upper neighbour of x is still contaminated and we have already reached $4(n-1)-1$.

Now let us consider *Down* propagation of the decontamination. The agent moves *Down* until reaching the upper neighbour of cell x. At this point, two neighbours of x are decontaminated and (n-2) time units have elapsed. To ensure propagation, only two directions for propagation are possible, *Left* and *Right*. *Down* and *Up* must be forbidden because they lead to an oscillation. Without loss of generality, let the agent move to the right (moving to the left produce symmetric propagation) and let movement to the left be forbidden. Now $f(\bullet,1,1,1,1) = (0,\Uparrow)$, and $f(\bullet,1,0,1,0) = (0,\Rightarrow)$ are permitted, while $f(\bullet,1,1,1,1) = (0,\Leftarrow)$, $f(\bullet,1,1,1,1) = (0,\Rightarrow)$, $f(\bullet,1,1,1,1) = (0,\Downarrow)$, $f(\bullet,1,0,1,0) = (0,\Leftarrow)$, $f(\bullet,1,0,1,0) = (0,\Uparrow)$ and $f(\bullet,1,0,1,0) = (0,\Downarrow)$ are forbidden.

The agent moves to the next column, and at this point it can move to one of the four directions. Clearly the movement to the left is forbidden because it leads to an oscillation, so possible movements are *Up*, *Down*, and *Right*. *Up* and *Down* movements will produce a symmetric propagation, so without loss of generality, let the *Down* movement be forbidden and the *Up* movement be permitted. Now we have two possible movements, *Up* and *Right*. Let us consider each possible movement and see what it will produce.

If the *Up* ($f(\bullet,0,1,1,1) = (0,\Uparrow)$) movement is permitted and *Down* and *Right* ($f(\bullet,0,1,1,1) = (0,\Downarrow)$, and $f(\bullet,0,1,1,1) = (0,\Rightarrow)$) are forbidden, the agent moves up to clean the column to the right of x. Then by the rules already present ($f(\bullet,1,1,1,1) = (0,\Uparrow)$, $f(\bullet,1,0,1,0) = (0,\Rightarrow)$, and $f(\bullet,0,1,1,1) = (0,\Uparrow)$), the agent requires another $(n-2)(n-1)+(n-2)$ time units to reach the left neighbours of the starting cell x. The time required is clearly greater than $4(n-1)-1$.

If the *Right* ($f(\bullet,0,1,1,1) = (0,\Rightarrow)$) movement is permitted and *Down* and *Up* moves ($f(\bullet,0,1,1,1) = (0,\Downarrow)$, and $f(\bullet,0,1,1,1) = (0,\Uparrow)$) are forbidden, the agent moves right to decontaminate the row situated above cell x. Then, by the rules already present ($f(\bullet,1,1,1,1) = (0,\Uparrow)$, $f(\bullet,1,0,1,0) = (0,\Rightarrow)$, and $f(\bullet,0,1,1,1) = (0,\Rightarrow)$) decontamination propagates to the column situated to the left of cell x. The time already elapsed is $2(n-1)$. At this point, two moves are possible, *Up* and *Down* (moves to *Right* and *Left* are forbidden because they produce an oscillation). Consider the case where *Up* is permitted and *Down* is forbidden. In this case, the agent moves *Up* to clean the column, and the time already elapsed is $3(n-1)$ time units. Then, the unique permitted movement left will keep x moving until reaching the column to the right of x, and the time already elapsed is $4(n-1)-2$ time units. At this point, two movement are possibles, *Right* and *Down*, both of which require more than two movement to reach the right neighbours of

x.

Now consider the case where the *Up* movement is permitted and *Down* is forbidden. Possible moves starting from x are *Down*, *Right* and *Up*, and possible rules are $(f(\bullet,0,1,1,1) = (0,\Downarrow),\ f(\bullet,1,0,1,1) = (0,\Downarrow),\ f(\bullet,1,0,1,0) = (0,\Rightarrow),\ f(\bullet,0,1,1,1) = (0,\Rightarrow),\ f(\bullet,0,1,0,1) = (0,\Uparrow), f(\bullet,1,1,0,0) = (0,\Uparrow),\ f(\bullet,1,0,0,0) = (0,\Leftarrow),\ f(\bullet,1,0,0,1) = (0,\Leftarrow))$. In this case, the agent moves *Up* to clean the column, and the time already elapsed is $3(n-1)$ time units. The unique permitted movement left will keep x moving until reaching the column at the right of x. The time elapsed is $4(n-1)-1$ and all the neighbours of x are decontaminated.

At this point the move will be repeated until all cells are decontaminated. The set of permitted moves corresponds to the set of rules given in Table 5.13, and the movement corresponds to the movement in Figure 5.11 or to a symmetric one. □

Configuration	Next State	Agents Movement
$\{\bullet,1,1,1,1\}$	0	$\Downarrow$
$\{\bullet,1,0,1,1\}$	0	$\Downarrow$
$\{\bullet,0,1,0,1\}$	0	$\Uparrow$
$\{\bullet,1,1,0,0\}$	0	$\Uparrow$
$\{\bullet,1,0,0,0\}$	0	$\Leftarrow$
$\{\bullet,1,0,0,1\}$	0	$\Leftarrow$
$\{\bullet,1,0,1,0\}$	0	$\Rightarrow$
$\{\bullet,0,1,1,1\}$	0	$\Rightarrow$
$\{\bullet,0,0,0,1\}$	0	$\Downarrow$
$\{\bullet,0,0,1,1\}$	0	$\Downarrow$
$\{\bullet,0,0,1,0\}$	0	$\Rightarrow$
$\{\bullet,0,1,1,0\}$	0	$\Rightarrow$
$\{\bullet,1,0,0,0\}$	0	$\Leftarrow$
$\{\bullet,1,0,0,1\}$	0	$\Leftarrow$
$\{\bullet,0,1,0,0\}$	0	$\Uparrow$
$\{\bullet,0,0,0,0\}$	0	*Terminate*

Table 5.9: Rule Table for Temporal Decontamination by a Single Agent in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood.

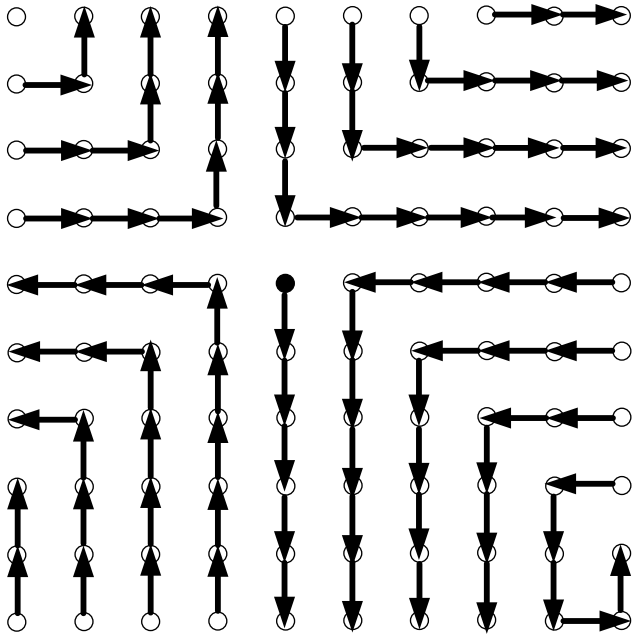

Figure 5.11: Temporal Decontamination by a Single Agent in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood

5.3.3.2 Two Agents

In the case of two agents, the agents are located in two adjacent columns and move in the same direction *Down* or *Up* cleaning the column, they then move to the next column, where they move in the same direction as in the previous column (*Down* or *Up*). Agents keep moving in the same direction during the whole decontamination process. An agent terminates when there is no more cells to decontaminate (i.e., all cells in its visibility range are decontaminated) (see Figure 5.12). The set of rules given in Table 5.12 show a possible solution to the decontamination.

Theorem 5.12. *With two agents, temporal decontamination in a circular mobile cellular automata is possible if the immunity time is* $\geq n + 1$.

Configuration	Next State	Agents Movement
$\{\bullet, 1, 1, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, \bullet, 0\}$	0	$\Leftarrow$
$\{\bullet, 1, 0, 0, 0\}$	0	$\Leftarrow$
$\{\bullet, \bullet, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, \bullet, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Terminate*

Table 5.10: Rule Table for Temporal Decontamination by Two Agents in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood.

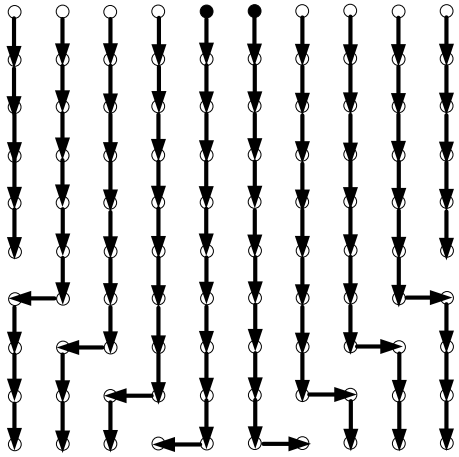

Figure 5.12: Temporal Decontamination by Two agents in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood

5.3.3.3 Three agent

With three agents, we use a similar solution but with an agent between the two agents used in the previous solutions (see Figure 5.13). The third agent has the only task of

decontaminating the column in the middle, and then it terminates. The set of rules given in Table 5.13 show a possible solution to the decontamination

Theorem 5.13. *With three agents, temporal decontamination in a circular mobile cellular automata is possible if the immunity time is* $\geq n + 1$.

Configuration	Next State	Agents Movement
$\{\bullet, 1, 1, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, \bullet, 0\}$	0	$\Leftarrow$
$\{\bullet, 1, 0, 0, 0\}$	0	$\Leftarrow$
$\{\bullet, \bullet, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, \bullet, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Terminate*

Table 5.11: Rule Table for Temporal Decontamination by Three Agents in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood.

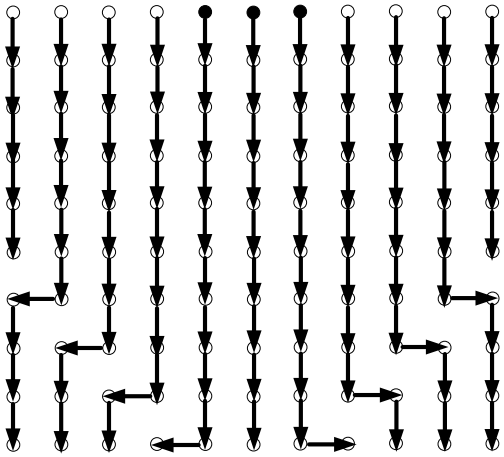

Figure 5.13: Temporal Decontamination by Three agents in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood

5.3.3.4 More than 3 Agents

We apply the same strategies as in the case of the finite mobile cellular automata (Strategy 1, Strategy 2), but by duplicating the number of agents and placing them in adjacent columns, this allows them to move in opposite directions (see Figure 5.14). The set of rules is given in Table 5.12, and these rules correspond to the movement of each group of siblings in opposite directions in two columns, turning right or left depending on the starting column as soon as they become adjacent. Note that this technique is possible only because the two siblings become adjacent but never move on the same cell, hence the need for placing them at an odd distance (i.e., separating them with an even number of cells).

Strategy 1: Siblings at an odd distance. This strategy applies when the distance between each two sibling is odd.

Theorem 5.14. *Let t_1 be the largest integer smaller than or equal to t such that $t_1 + 1$ divides n and let t_1 be odd. Temporal decontamination can be achieved in a circular mobile cellular automata with a Von Neumann neighbourhood and an immunity time t using $k = 2\frac{2n}{(t_1+1)}$ agents.*

Configuration	Next State	Agents Movement
$\{\bullet, \bullet, \bullet, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 1, \bullet, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, \bullet, \bullet\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 1, 0\}$	0	$\Uparrow$
$\{\bullet, 1, 1, \bullet, 0\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 1, \bullet\}$	0	$\Uparrow$
$\{\bullet, 1, 0, \bullet, \bullet\}$	0	$\Leftarrow$
$\{\bullet, 1, \bullet, \bullet, 0\}$	0	$\Leftarrow$
$\{\bullet, \bullet, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, \bullet, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 1, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 1, 1, \bullet\}$	0	$\Uparrow$
$\{\bullet, 1, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, \bullet, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 1, \bullet, 0, 0\}$	0	$\Leftarrow$
$\{\bullet, 1, 0, 0, \bullet\}$	0	$\Leftarrow$
$\{\bullet, 0, \bullet, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, \bullet, 0\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, \bullet, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, \bullet, 0\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, 0\}$	0	*Terminate*

Table 5.12: Rule Table for Temporal decontamination by Multiple Agents in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood: Strategy 1.

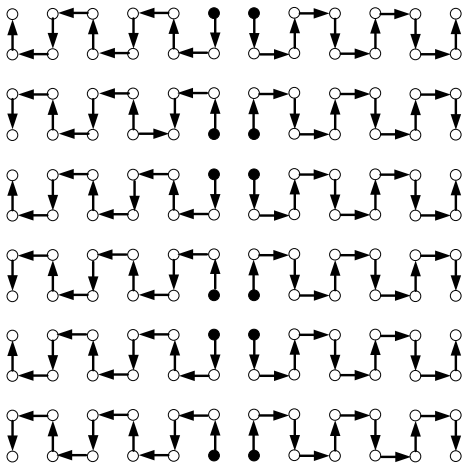

Figure 5.14: Temporal Decontamination by Multiple Agents in a Circular Mobile Cellular Automata with the Von Neumann neighbourhood: Strategy 1.

Strategy 2: Siblings at an even distance. This strategy is applied when the distances between two siblings is even.

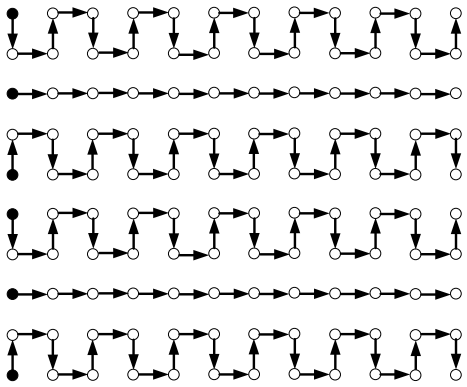

Figure 5.15: Temporal Decontamination by Multiple Agents in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood: Strategy 2.

Theorem 5.15. *Let t_2 be the largest integer smaller than or equal to t such that $t_2 + 2$ divides n and let t_2 be even. Temporal decontamination can be achieved in a circular mobile cellular automata with the Von Neumann neighbourhood and an immunity time t using k agents where $k = 2\frac{3n}{(t_2+2)}$.*

Configuration	Next State	Agents Movement
$\{\bullet, \bullet, \bullet, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 1, \bullet, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, \bullet, \bullet\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 1, \bullet\}$	0	$\Uparrow$
$\{\bullet, 1, 0, \bullet, \bullet\}$	0	$\Leftarrow$
$\{\bullet, 1, \bullet, \bullet, 0\}$	0	$\Leftarrow$
$\{\bullet, 1, \bullet, \bullet, \bullet\}$	0	$\Leftarrow$
$\{\bullet, \bullet, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, \bullet, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, \bullet, \bullet, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 1, 1, \bullet, 0\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 1, 0\}$	0	$\Uparrow$
$\{\bullet, 1, 0, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 1, 1, 0\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, \bullet, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 1, \bullet, 0, \bullet\}$	0	$\Leftarrow$
$\{\bullet, 1, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 1, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 1, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, \bullet, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, \bullet, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, \bullet, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 1, \bullet, 0\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, \bullet, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, \bullet, 0\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, 0\}$	0	*Terminate*
$\{\bullet, 0, \bullet, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, 0, \bullet\}$	0	*Terminate*
$\{\bullet, 0, \bullet, 0, \bullet\}$	0	*Terminate*

Table 5.13: Rule Table for Temporal decontamination by Multiple Agents in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood: Strategy 2.

5.3.4 Circular Mobile Cellular Automata with the Moore Neighbourhood

If we enlarge visibility to include the diagonal neighbours (see Figure 5.16), we obtain a solution which is valid for any number of agents ≥ 2 (solutions for single, two, and three agents are similar to the one shown for the case of a Von Neumann neighbourhood). The difference with the Von Neumann case is that we can avoid fusing two agents when they meet in the same cell. The set of rules given in Table 5.14 achieves decontamination in a circular mobile cellular automata with temporal immunity.

Theorem 5.16. *Temporal decontamination can be achieved in a finite mobile cellular automata with a Moore neighbourhood and an immunity time t (where $t \geq 0$) using k agents where:*

$$k = \begin{cases} 2 \times \frac{2n}{t+1} & \textit{if} \quad 2n \mod t+1 = 0 \\ 2 \times \lfloor \frac{2n}{t+1} \rfloor + 3 & \textit{Otherwise} \end{cases}$$

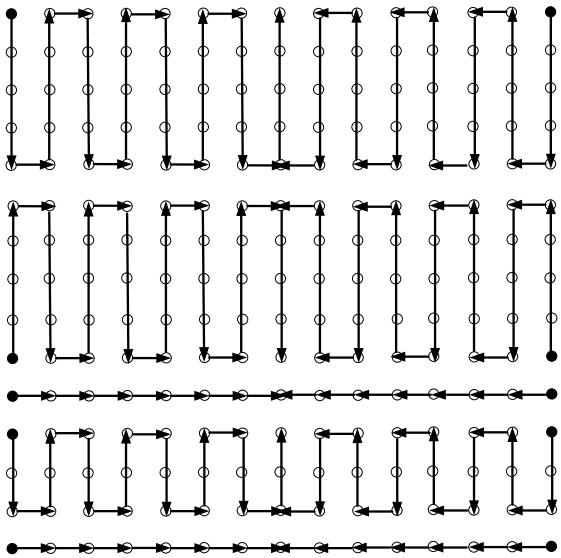

Figure 5.16: Rule Table for Temporal Decontamination by Multiple Agents in a Circular Mobile Cellular Automata with the Moore Neighbourhood

Configuration	Next State	Agents Movement
$\{\bullet, 1, 1, \bullet, \bullet, \bullet, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, 0, 0, \bullet, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, \bullet, 0, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, 0, 0, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, \bullet, \bullet, \bullet, \bullet, \bullet, 1\}$	0	$\Leftarrow$
$\{\bullet, 1, 1, 0, 0, \bullet, \bullet, \bullet, 1\}$	0	$\Leftarrow$
$\{\bullet, 1, 1, \bullet, \bullet, \bullet, 0, 0, 1\}$	0	$\Leftarrow$
$\{\bullet, 1, 1, 1, 1, \bullet, \bullet, \bullet, 1\}$	0	$\Uparrow$
$\{\bullet, 1, 1, 1, 1, \bullet, 0, 0, 1\}$	0	$\Uparrow$
$\{\bullet, 1, 1, 1, 0, 0, 0, \bullet, 1\}$	0	$\Uparrow$
$\{\bullet, 1, 1, 1, 0, 0, 0, 0, 1\}$	0	$\Uparrow$
$\{\bullet, 1, 1, 0, 0, 0, 0, \bullet, 1\}$	0	$\Leftarrow$
$\{\bullet, 1, 1, \bullet, 0, 0, 0, 0, 1\}$	0	$\Leftarrow$
$\{\bullet, 1, 1, \bullet, 0, 0, 0, \bullet, 1\}$	0	$\Leftarrow$
$\{\bullet, 1, \bullet, 0, 0, 0, 0, \bullet, 1\}$	0	$\Leftarrow$
$\{\bullet, \bullet, \bullet, \bullet, 1, 1, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, 0, 0, 1, 1, 1, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, \bullet, 1, 1, 1, 1, 0\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 1, 1, 1, 1, 0\}$	0	$\Downarrow$
$\{\bullet, \bullet, 0, 0, 1, 1, 1, \bullet, \bullet\}$	0	$\Rightarrow$
$\{\bullet, \bullet, \bullet, \bullet, 1, 1, 1, 0, 0\}$	0	$\Rightarrow$
$\{\bullet, \bullet, \bullet, \bullet, 1, 1, 1, \bullet, \bullet\}$	0	$\Rightarrow$
$\{\bullet, \bullet, 1, 1, 1, 1, 1, \bullet, \bullet\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 1, 1, 1, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 1, 1, 1, \bullet, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 1, 1, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 0, 1, 1, 1, \bullet, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, \bullet, 1, 1, 1, 0, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, \bullet, 1, 1, 1, \bullet, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, \bullet, 1, 1, \bullet, 0, 0\}$	0	$\Rightarrow$
$\{\bullet, \bullet, 1, 1, 0, 0, 0, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 1, \bullet, 0, 0, 0\}$	0	$\Uparrow$
$\{\bullet, \bullet, 1, 1, 0, 0, 0, \bullet, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 1, \bullet, \bullet, \bullet, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, \bullet, \bullet, \bullet, 1, 1, 0\}$	0	$\Downarrow$
$\{\bullet, 0, 0, 0, 0, \bullet, 1, 1, 0\}$	0	$\Downarrow$
$\{\bullet, \bullet, \bullet, \bullet, 0, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, \bullet, 0, 0, 0, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 0, \bullet, \bullet, \bullet, \bullet, \bullet, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, 0, 0, \bullet, \bullet, \bullet, 0\}$	0	*Terminate*
$\{\bullet, 0, 0, \bullet, \bullet, \bullet, 0, 0, 0\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, \bullet, 0, 0, 0, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, \bullet, 0, 0, 0, 0, 0, \bullet, \bullet\}$	0	*Terminate*
$\{\bullet, \bullet, \bullet, \bullet, 0, 0, 0, 0, 0\}$	0	*Terminate*

Table 5.14: Rule Table for Temporal Decontamination by Multiple Agents in a Circular Mobile Cellular Automata with the Moore Neighbourhood.

5.4 Temporal Decontamination with Cloning

While basic decontamination with a Von Neumann neighbourhood is quite simple and the possibility of cloning does not really have an impact on the solution, with temporal decontamination we can observe that cloning indeed makes a difference. Two agents are not allowed in the same cell; if they happen to move on the same cell, they fuse into one. If cloning is not allowed and we need the number of agents to be fixed and pre-determined, the set of rules must forbid two agents from moving on the same cell. Note that this might not always be possible because of the limited visibility of an agent. In fact, if two agents are separated by one cell, they cannot see each other, and they cannot avoid fusing in that cell if they move towards each other. The case when cloning is allowed is different as it might allow fusing some agents because they can be regenerated during the life of the algorithm. In this section we study the impact of this capability on the decontamination problem.

5.4.1 Finite Mobile Cellular Automata

In this section, we consider the more general case when the local transition function can return several directions. As discussed in Section 1, in the case of basic decontamination the result is the same as in the case of basic decontamination without cloning. We now consider the case of temporal decontamination in a finite mobile cellular automata with $t > 1$ and Von Neumann neighbourhood.

Initial Placement. We divide the n cells of the first column in groups of at most $t + 1$ consecutive cells and employ two agents (the siblings) in each group, one on top, and one on the bottom. The two siblings can be separated by an arbitrary distance smaller than t.

Pattern of Movement. The set of rules is given in Table 5.15 and it corresponds to the movement of each group of siblings in opposite directions in the column. Agents turn right once they become adjacent (see cells b and c in Figure 5.17), or when they meet in the same cell (see cell a in Figure 5.17), and in this case they will fuse and act as a single agent. If they fuse, when the agent is in the next column and realizes that its *up* and *down* neighbours are contaminated, it will duplicate and propagate in both directions.

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 1, 1, 1\}$	0	$\Downarrow \Uparrow$
$\{\bullet, 0, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, \bullet, 0, \bullet\}$	0	*Stop*
$\{\bullet, 0, \bullet, 0, 0\}$	0	*Stop*
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Stop*
$\{\bullet, 0, 0, 0, 0\}$	0	*Stop*

Table 5.15: Rule Table for Temporal Decontamination by Multiple Agents with Cloning Capability in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood.

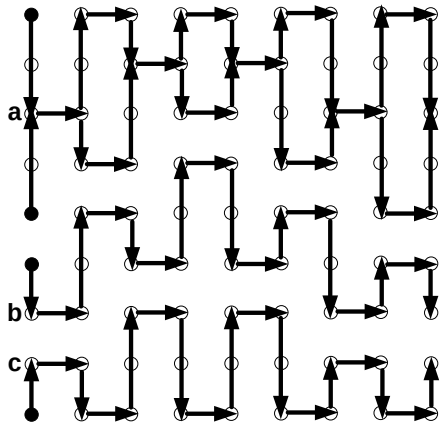

Figure 5.17: Temporal Decontamination by Multiple Agent with Cloning capability in a Finite Mobile Cellular Automata with the Von Neumann Neighbourhood.

Theorem 5.17. *Temporal decontamination can be achieved in a finite mobile cellular automata with a Von Neumann neighbourhood and an immunity time t using $k = \lceil \frac{2n}{t} \rceil$ agents that can clone.*

Proof. We now show that, by following the rules of Table 5.15, decontamination is achieved monotonically. That is, once *decontaminated*, every cell stays clean until the end of the process, when all cells are decontaminated.

Let us call *entry points* of decontamination in a column the cells in such a column that become *active* due to a left *active* neighbour (horizontal propagation). We now prove by induction using the number of columns that for each column i there is a time t when: (i) all cells in column i are either *decontaminated* or *active*; (ii) all cells in column $i-1$ (for $i > 0$) are *decontaminated*; (iii) by time $t_1 + t$ all right neighbours of a *decontaminated* cell of column i are either *decontaminated* or *active*; and (iv) the distance between any two consecutive entry points in column $i+1$ (for $i < n-1$) is smaller than t.

1. Base - column 0: According to the set of rules (Table 5.15), the *active* state propagates vertically in both directions in the first column (see rules $f(\{\bullet, 0, 0, 1, 1\}) = (0, \Downarrow)$, $f(\{\bullet, 0, \bullet, 1, 1\}) = (0, \Downarrow)$, $f(\{\bullet, 0, 1, 1, \bullet\}) = (0, \Uparrow)$ and $f(\{\bullet, 0, 1, 1, 0\}) = (0, \Uparrow)$). Since the maximal distance between initially consecutive active cells is t by construction, within $\lfloor \frac{t-1}{2} \rfloor$ time units all the cells on column 0 are either *active* or *decontaminated*. According to the local rules, decontamination then propagates to column 1. In fact, by rule $f(\{\bullet, 0, 1, 1, 1\}) = (0, \Uparrow\Downarrow)$ if there is a single *active* cell with up and down *contaminated* neighbours, it will duplicate and start propagating in both direction. If instead there is a pair of *active* cells, each of them starts propagating in the direction of their *contaminated* neighbours in the column (rules: $f(\bullet, 0, \bullet, 1, 1) = (0, \Downarrow)$ and $f(\bullet, 0, 1, 1, \bullet) = (0, \Uparrow)$). Since the propagation in column 1 happens for all *active* cells within $\lfloor \frac{t-1}{2} \rfloor$ time units from the beginning, the distance between any two consecutive entry points in column 1 cannot be greater than t. By a similar argument as that for column 0, all cells in column 1 will then become *decontaminated* or *active* within other $\lfloor \frac{t-1}{2} \rfloor$ time units. Thus, within at most t time units, all the cells in column 0 and in column 1 will be either *decontaminated* or *active*. We can conclude that the base case is true.

2. Induction hypothesis: Assume that there is a time when all cells of column i $(0 < i < n-1)$ and all their right neighbours in column $i+1$ are either in an *active* or *decontaminated* state, their left neighbours in column $i-1$ are clean, and the entry points in column $i+1$ are at a distance of at most t.

3. Induction Step: Consider column $i+1$. By the induction hypothesis we know that there is a time t when all cells in column $i-1$ are clean, the cells in columns i and $i+1$

are either *active* or *decontaminated*, and the entry points in column $i+1$ are at maximum distance t from each other. It follows that the decontamination propagates vertically in column $i+1$ within $\lfloor \frac{t-1}{2} \rfloor$ time units from time t, thus leaving all cells in column i clean. Decontamination propagates then to column $i+2$ and within other $\lfloor \frac{t-1}{2} \rfloor$ time units all the cells in the column $i+2$ become either *decontaminated* or *active*. We can conclude that by time $t_1 + t$ all cells in column $i+1$ together with all their right neighbours are either *decontaminated* or *active* and the cells in column i are clean, thus concluding the proof. □

5.4.2 Circular Mobile Cellular Automata

In this section we consider the more general case when the local transition function can return several directions. As discussed in Section 1, in the case of basic decontamination the result is the same as in the case of basic decontamination without cloning. We now consider the case of temporal decontamination in a circular mobile cellular automata with $t > 1$ and Von Neumann neighbourhood.

Initial Placement. We divide n cells of any column of the circular mobile cellular automata in groups of at most $t+1$ consecutive cells and employ two agents (the siblings) in each group, one on top, and one on the bottom. The two siblings can be separated by an arbitrary distance smaller than T.

Pattern of Movement. The set of rules is given in Table 5.16 and corresponds to the movement of each group of siblings in opposite directions in the column. Agents turn in both directions once they become adjacent (see cells b and c in Figure 5.18), or when they meet in the same cell (see cell a in Figure 5.18), and in this case they will fuse and act as a single agent. If they fuse, when the agent is in the next column and realizes that its *up* and *down* neighbours are contaminated, it will duplicate and propagate in both directions.

Configuration	Next State	Agents Movement
$\{\bullet, 0, 0, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 1, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 1, 0\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 1, 1, 1\}$	0	$\Downarrow \Uparrow$
$\{\bullet, 0, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 0, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 0, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 0, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 0, 0, 1, \bullet\}$	0	$\Rightarrow$
$\{\bullet, 0, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 0, 1, 0\}$	0	$\Rightarrow \Leftarrow$
$\{\bullet, 1, \bullet, 1, 0\}$	0	$\Rightarrow \Leftarrow$
$\{\bullet, 1, 0, 1, \bullet\}$	0	$\Rightarrow \Leftarrow$
$\{\bullet, 1, 1, 0, \bullet\}$	0	$\Uparrow$
$\{\bullet, 1, \bullet, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, 0, 1\}$	0	$\Downarrow \Uparrow$
$\{\bullet, 0, \bullet, 1, 0\}$	0	$\Rightarrow$
$\{\bullet, 1, 0, 0, 1\}$	0	$\Downarrow$
$\{\bullet, 1, 1, 0, 0\}$	0	$\Uparrow$
$\{\bullet, 1, 0, 0, \bullet\}$	0	$\Leftarrow$
$\{\bullet, 1, \bullet, 0, 0\}$	0	$\Leftarrow$
$\{\bullet, 1, 0, 0, 0\}$	0	$\Leftarrow$
$\{\bullet, 0, \bullet, 0, \bullet\}$	0	*Stop*
$\{\bullet, 0, \bullet, 0, 0\}$	0	*Stop*
$\{\bullet, 0, 0, 0, \bullet\}$	0	*Stop*
$\{\bullet, 0, 0, 0, 0\}$	0	*Stop*
$\{\bullet, 0, 0, \bullet, \bullet\}$	0	*Stop*
$\{\bullet, 0, \bullet, \bullet, 0\}$	0	*Stop*
$\{\bullet, \bullet, \bullet, 0, 0\}$	0	*Stop*
$\{\bullet, \bullet, 0, 0, \bullet\}$	0	*Stop*

Table 5.16: Rule Table for Temporal Decontamination by Multiple Agents with Cloning Capability in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood.

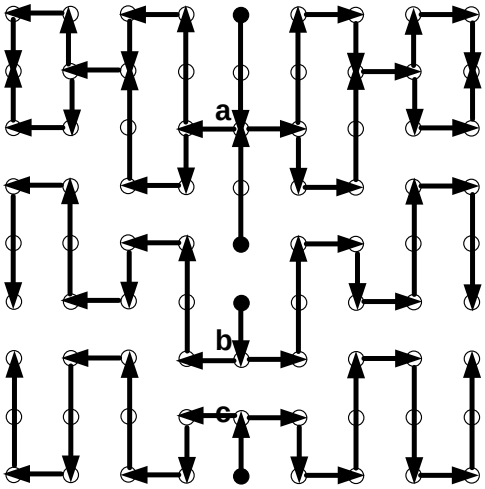

Figure 5.18: Temporal Decontamination by Multiple Agents with Cloning Capability in a Circular Mobile Cellular Automata with the Von Neumann Neighbourhood.

The proof of the following theorem is similar to the previous one.

Theorem 5.18. *Temporal decontamination can be achieved in a circular mobile cellular automata with a Von Neumann neighbourhood and an immunity time* t *using* $K = 2\times\lceil\frac{2n}{t}\rceil$ *agents that can clone.*

5.5 Conclusion

In this chapter, we have continued the line of investigation started in the previous chapter to study the network decontamination problem in cellular systems where the spread of faults, as well as the decontamination process, are regulated by local rules. We have focused on a two-dimensional three-state mobile cellular automata with and without temporal immunity, to look at the impact that *active* cells, in an otherwise reactive environment, have on the decontamination problem. We studied finite and circular mobile cellular automata with Von Neumann and Moore neighbourhoods. In each case we have described local rules that achieves decontamination. We have measured the efficiency of our sets of rules by considering the number of agents required to decontaminate the whole network, and with respect to this metric, we have shown some optimal results. We have also introduced the concept of the cloning and fusing capability of an agent and have shown the extra power that gives to a solution.

We have described rules achieving optimal basic decontamination and prove their optimality in the case of finite and circular mobile cellular automata with and without cloning and Von Neumann neighbourhood. A solution is given for the temporal decontamination in the case of both Von Neumann and Moore neighbourhoods, the effect of the enlarged neighbourhood is that we can avoid the meeting of two agents at the same cell and we can develop a more general solution. In the case of mobile circular automata with temporal immunity without cloning, we describe four different strategies for the cases of, one, two, three and $k \geq 4$ agents. Decontamination is also studied in the case where agents have cloning capability for finite and circular mobile cellular automata. Table 5.17 summarizes the results obtained in this chapter.

In this chapter we omitted the proofs of several theorems that are easy consequences of the definition of the presented strategies.

	Number of Agents	Immunity Time
Decontamination without Cloning		
Finite Mobile Celular Automata		
Von Neumann Neighbourhood	$k = n$	$t = 1^*$
	$k = 1$	$t = 2n - 1^*$
	$k = \min\{\frac{2n}{t_1+1}, \frac{3n}{t_2+1}\}$	t
Moore Neighbourhood	$k = \frac{2n}{t+1}$ If $2n \mod (t+1) = 0$ $k = \lfloor \frac{2n}{t+1} \rfloor + 2$ Otherwise	t
Circular Mobile Celular Automata		
Von Neumann Neighbourhood	$k = 2n$	$t = 1^*$
	$k = \min\{2\frac{2n}{t_1+1}, 2\frac{3n}{t_2+1}\}$	t
	$k = 1$	$t = 4(n-1) - 1^*$
	$k = 2, 3$	$t = (n+1)$
Moore Neighbourhood	$k = 2 \times \frac{2n}{t+1}$ If $2n \mod (t+1) = 0$ $k = 2 \times \lfloor \frac{2n}{t+1} \rfloor + 3$ Otherwise	t
Decontamination with Cloning		
Finite Mobile Celular Automata		
Von Neumann Neighbourhood	$k = \lceil \frac{2n}{t} \rceil$	t
	$k = n$	$t = 1^*$
Circular Mobile Celular Automata		
Von Neumann Neighbourhood	$k = 2\lceil \frac{2n}{t} \rceil$	t
	$k = 2n$	$t = 1^*$

Table 5.17: Chapter 5: Summary of Results

*The proposed strategy is an optimal solution

Chapter 6

Decontamination by Mobile Agents

In this chapter we consider decontamination of mesh and torus by Mobile Agents (MA). We consider a network that has been contaminated by a persistent and active virus. When infected, a network site will continuously attempt to spread the virus to its neighbours. Similarly to the case of chapters 4 and 5, at any time nodes can be contaminated, decontaminated (clean), or guarded (if they contain at least an agent). All nodes are initially contaminated except for one (the homebase) where a team of mobile agents is located. Agents can move in the network from a node to a neighbouring node and a contaminated node is transformed into decontaminated when an agent it passes by. However, once the cleaner departs, the decontaminated node can be recontaminated by contaminated neighbours. As in the models already discussed, when immunity time $t \geq 0$ and once the cleaner departs, the decontaminated node is immune for t time units to viral attacks from infected neighbours. After the immunity time t is elapsed recontamination can occur.

In this chapter we study the decontamination problem using a team of mobile agents with temporary immunity $t(t \geq 0)$. Contrary to the model presented in the previous chapter, agents do not have visibility capability; instead an agent can read the whiteboard and write messages onto the whiteboard to communicate with other agents. The agent has its own memory to store information obtained from the whiteboard, and to remember past performed actions. We propose solutions to the decontamination of a mesh and torus, we prove their correctness, and we propose and prove optimality of a solution to the mesh decontamination with temporal immunity by a single agent.

6.1 Mesh Decontamination

In this section we describe two general strategies for the decontamination of a mesh with temporary immunity t by a team of mobile agents. We prove their correctness, and we show optimality for the case of a single agent.

6.1.1 Multiple Mobile Agents

Algorithm 1.

The algorithm consists of two phases: the initialization phase, and the decontamination phase.

In the initialization phase, we place the$\lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$ agents equidistant in the first column. The first agent is placed on the node $P(0,0)$, the second agent is placed on node $P(\lceil \frac{t}{2} \rceil, 0)$, and the next $\lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil - 2$ agents are placed equidistant leaving $\lceil \frac{t}{2} \rceil + 1$ nodes in between each pair. Notice that the distance between the last agent and the lower left corner might be $\leq \lceil \frac{t}{2} \rceil$. In this way (see Figure 6.1), we create $\lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil - 1$ equal intervals, while the last interval might be smaller than the previous intervals.

In the decontamination phase, agents move synchronously. Each agent moves to the $\lceil \frac{t}{2} \rceil$th contaminated nodes from itself on the first column, then moves to the next column, moves back along the same column for $\lceil \frac{t}{2} \rceil$ nodes, and then moves to the next column again. This process is repeated until agents reach the last column and clean it. Each iteration consists then of the following moves: SOUTH ($\Downarrow$), EAST ($\Rightarrow$), NORTH ($\Uparrow$), and WEST($\Leftarrow$) (see Figure 6.1).

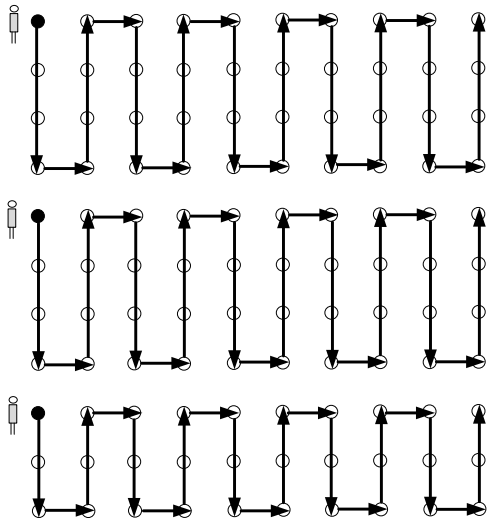

Figure 6.1: Mesh Decontamination by Multiple Mobile Agents with Temporal Immunity: Strategy 1

Algorithm 1 Mesh Decontamination with Temporal Immunity: Strategy 1

input: A contaminated $m \times n$-Mesh ($m < n$) with temporary immunity t and A agents

output: The given mesh is decontaminated

Variables:

$|A| = \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$, $i, j, k, P(i,j), MCount_k$

InitializationMesh$(P(i,j), |A|)$

DecontaminationMesh$(P(i,j), |A|, MCount_k)$

function InitializationMesh$(P(i,j), |A|)$
  All agents are in node $P(0,0)$
  for do$(k = 1, k \leq |A|, k++)$
    for $(i = 0, i \leq m-1, i = i + \lceil \frac{t}{2} \rceil)$ **do**
      Move a_k to the node $P(i,0)$
    end for
  end for
end function

```
function DecontamintionMesh(P(i,j), |A|, MCount_k)
    for (j = 0, j < n, j + 2) do
        for (k = 1, k ≤ |A|, K++ do
            MCount_k = 0
            for (i = 0, i < ⌈t/2⌉, i++) do
                Move(i + 1, j)
                MCount_k++
            end for
            if (MCount_k = ⌈t/2⌉ − 1) then
                Move(i, j + 1)
            else
                Wait (⌈t/2⌉ − MCount_k)
                Move(i, j + 1)
            end if
            MCount_k = 0
            for (i = ⌈t/2⌉ − 1, i > 0, i − −) do
                Move(i − 1, j)
                MCount_k++
            end for
            if (MCount < ⌈t/2⌉ − 1) then
                Move (i,j+1)
            else
                Wait ((⌈t/2⌉ − MCount_k))
                Move(i,j+1)
            end if
        end for
    end for
end function
```

Theorem 6.1. *A mesh with temporal immunity t can be decontaminated by* $|A| = \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$ *agents.*

Proof. To prove the correctness of Algorithm 1 presented above, we need to prove that the decontamination is achieved monotonically and all nodes will be decontaminated by the end of the decontamination process. That is, once *decontaminated*, every node

stays *decontaminated* until the end of the decontamination process, when all cells are *decontaminated*. We now prove by induction using the number of columns that for each column i there is a time t when: (i) all nodes in column i are either decontaminated or contain an agent; (ii) all nodes in column $i-1$ (for $i>0$) are decontaminated; and (iii) by time t all right neighbours of a decontaminated node of column i are either decontaminated or contain an agent.

1. Base Case: Consider the columns 0 and 1. At time $t=0$ all agents are in node $P(0,0)$. In the initialization phase, agents move synchronically to their location with the maximum distance between any two agent being $\lceil \frac{t}{2} \rceil$. Agents move synchronically according to the following pattern: move SOUTH for $\lfloor \frac{t}{2} \rfloor$ moves, then move WEST for 1 move then NORTH for $\lfloor \frac{t}{2} \rfloor$. The total time would be at most t, so column 0 is decontaminated within $\lfloor \frac{t}{2} \rfloor$ time units and never becomes recontaminated, and all nodes in column 1 are either decontaminated or contain an agent within t time units. The base case is true.

2. Assumption Step: The columns i and $i+1$. Let us assume that at step i $(i>0)$, all nodes in column i are either decontaminated or contain an agent, and all nodes in column $i-1$ (for $i>0$) are decontaminated. By time t all right neighbours (column $i+1$) of a decontaminated node in column i are either decontaminated or contain an agent, and once contaminated a node in column $i-1$ never become recontaminated.

3. Induction Step: As defined in algorithm 1, at decontamination step $i+1$, all agents move according to the following pattern: SOUTH or NORTH for $\lfloor \frac{t}{2} \rfloor$; move WEST for 1 move; and NORTH or SOUTH for $\lfloor \frac{t}{2} \rfloor$, which in total takes at most t time units and lets the agents move by all nodes in column $i+1$ and $i+2$. Furthermore, by the assumption step, nodes decontaminated in column i will never become recontaminated.

By the above proof, we have shown that the decontamination result is monotone and all nodes will be decontaminated. Therefore, we have proven that the mesh decontamination can be done by $\lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$ agents. □

Algorithm 2.

Algorithm 2 is another solution for decontaminating a mesh with temporary immunity t ($t \geq 1$) by a team of mobile agents. Initially all agents are placed on the node $P(0,0)$ (homebase). Agents have distinct identifiers between 1 and k (where k is the number of agents). Let id_a be the identifier of agent a. Each agent a waits for $t \times id_a$ time units and then moves in a snake-like pattern, going south until reaching the lower border, then west, then north until reaching the upper border and so on. While moving in such a pattern, the agents maintain a distance of t between themselves, in such a way that a node which is decontaminated for the first time at time t will then be traversed by an agent every t time units until it is permanently protected (see Figure 6.2).

Algorithm 2 Mesh Decontamination with Temporal Immunity: Strategy 2

input: A contaminated $m \times n$-Mesh ($m < n$) with temporary immunity t and A agents

output: The given mesh is decontaminated

Variables:

$|A| = \lceil \frac{2m-1}{t} \rceil$, $i, j, P(i,j)$

All agents are in node $P(0,0)$

Decontamination:

for $(k = 0, k < |A|, k++)$ **do**
  Wait $(K \times t)$
  for $(j = 0, j \leq n-2, j++)$ **do**
    for $(i = 0, i \leq n-2, i++)$ **do**
      Move to $P(i+1, j)$
    end for
    Move to $P(i, j+1)$
    for $(i = m-1, i \geq 1, i--)$ **do**
      Move to $P(i-1, j)$
    end for
    Move to $P(i, j+1)$
  end for
end for

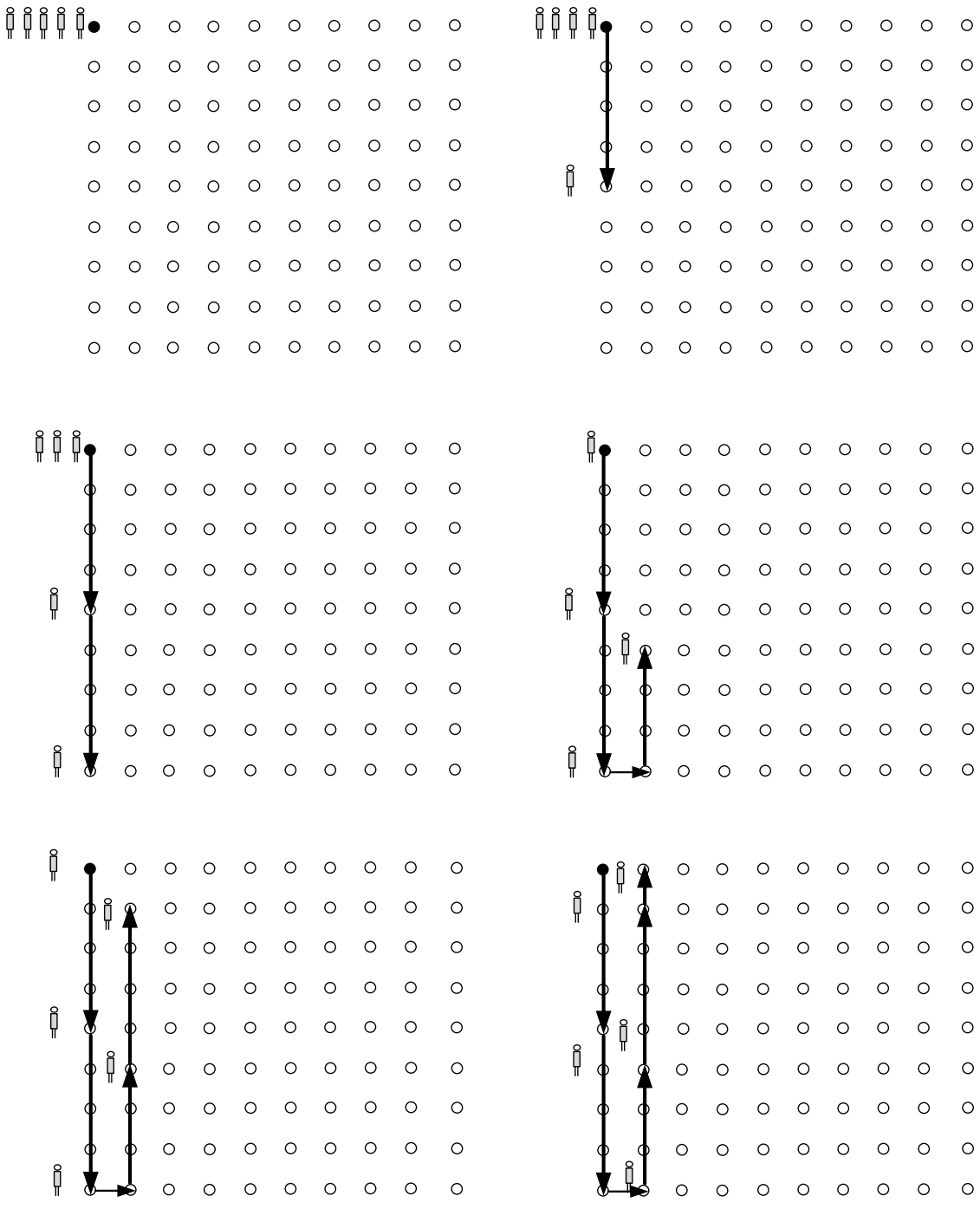

Figure 6.2: Mesh Decontamination by Multiple Mobile Agents with Temporal Immunity: Strategy 2

Theorem 6.2. *A mesh with temporal immunity t can be decontaminated by $|A| = \lceil \frac{2m-1}{t} \rceil$ agents.*

Proof. To prove the correctness of the algorithm, we need to prove that once decontaminated a node will never be recontaminated, and that all the nodes will be decontaminated by the end of the decontamination process. By construction a node $P(i,j)$, after being decontaminated for the first time, is then decontaminated again by an agent every t time units, thus remaining decontaminated until the visit of the last agent. Since the total number of agents passing by $P(i,j)$ is $\lceil \frac{2m-1}{t} \rceil$, by the time the last agent visits $P(i,j)$, the first agent that passed by $P(i,j)$ has reached the node at distance $(2m-1-t)$ on the snake-like path, which is the node $P(i-t, \lfloor \frac{i-t+2m-1}{m} \rfloor + j)$. Within t time units then, the first agent will have reached the node which is the right neighbour of $P(i,j)$, and $P(i,j)$ will be clean with all its neighbours clean. Since, by construction, the snake-path visits all the nodes, the proof follows.

□

With a single agent, both algorithms have the same required time immunity ($t = 2m-1$), and the same pattern of agent movement (see Figure 6.3). In the following, we prove optimality of the solutions in the case of a single agent.

Theorem 6.3. *Consider a $m \times n$-Mesh ($m < n$). When $|A| = 1$, the immunity time must be at least $2(m-1)+1$, and we have a unique optimal algorithm that decontaminates the whole mesh with immunity time $2(m-1)+1$.*

Proof. Independently from the strategy adopted to decontaminate the mesh, all the nodes must be decontaminated. Without loss of generality, let us consider the node $P(m-1,0)$. At some step of the strategy $P(m-1,0)$ must be decontaminated.

Claim 6.4. *If the path taken by the agent a is not the vertical one, then a should visits all columns before $P(m-1,0)$. Therefore, it should visit all nodes before $P(m-1,0)$.*

Proof. Consider $d(A,B)$ the minimum distance between A and B in the mesh, C_a the path taken by the agent a, and $C_a(M,N)$ the path between M and N. If $|C_a(M,N)| = k \geq m$, then all neighbours of N must be decontaminated in the subgraph $C_a(P(0,0),N)$. Consider $C'_a = C_a(M,N)$ does not contain nodes from the column $(i+1)$. Take the smallest such i, largest j in the column visited by C'_a. If the agent goes to node $P(m,0)$ to decontaminate it, then the distance traversed by a since it left the first column is $(d(P(i-k,0), P(i,j) + d(P(j,i), P(m-1,0)))$ where $(k \leq i)$, which is clearly greater

then m. To go back to the node $P(0, j+1)$, the same number of steps is required. The immunity time is $2(m-1)+1$, so the node $P(0, m-1)$ will be recontaminated. The only way to achieve decontamination is to visit all the nodes before the agent reach $P(m-1, 0)$. □

Claim 6.5. *Agent a must complete the decontamination of all columns $C_a - P(m-1, 0)$ before decontaminating any node in column C_{n-1}.*

Proof. Suppose C_a are not completely decontaminated and a node $P(i, n-1)$ remains contaminated. Consider the largest j, such that $P(j, 0)$ is decontaminated after the node $P(i, n-1)$. Going back and forth between $P(j+1, 0)$ and $P(i, n-1)$ will take more than the immunity time $2(m-1)+1$. Therefore, the node $P(j, 0)$ will be contaminated again before its neighbour $P(j+1, 0)$ is decontaminated. □

We supposed that all nodes must be visited before $P(m-1, 0)$. Moreover, the node $P(n-2, 0)$ is visited before any node $P(i, n-1)$ for every i, thus the path $P(n-2, 0) \rightarrow P(i, n-1) \rightarrow P(m-1, 0)$. □

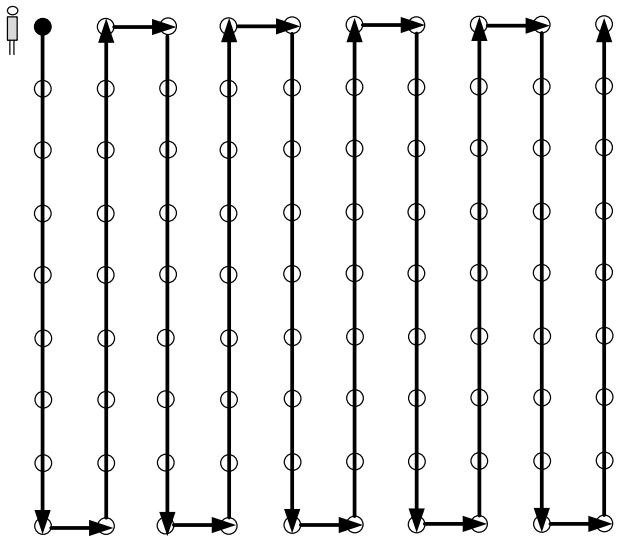

Figure 6.3: Mesh Decontamination by a Single Mobile Agent with Temporal Immunity

6.2 Torus Decontamination

In this section we describe and prove the correctness of two general strategies for the decontamination of a torus with temporary immunity t by a team of mobile agents.

To decontaminate a torus we can employ two algorithms (see Algorithms 3 and 4) based on Algorithm 1 and Algorithm 2 described in section 6.1. Instead of starting with $\lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$ agents for the first algorithm, and $\lceil \frac{2m-1}{t} \rceil$ agents for the second algorithm, we start with $2\lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$ agents and $2\lceil \frac{2m-1}{t} \rceil$ agents respectively in two adjacent columns. We let them move as in Algorithm 1 and Algorithm 2, but in opposite directions (see Figures 6.4 and 6.5). The proof of correctness follows the same lines as the ones of section 6.1.

Theorem 6.6. *A torus with temporal immunity t can be decontaminated by $|A| = 2\lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$ agents.*

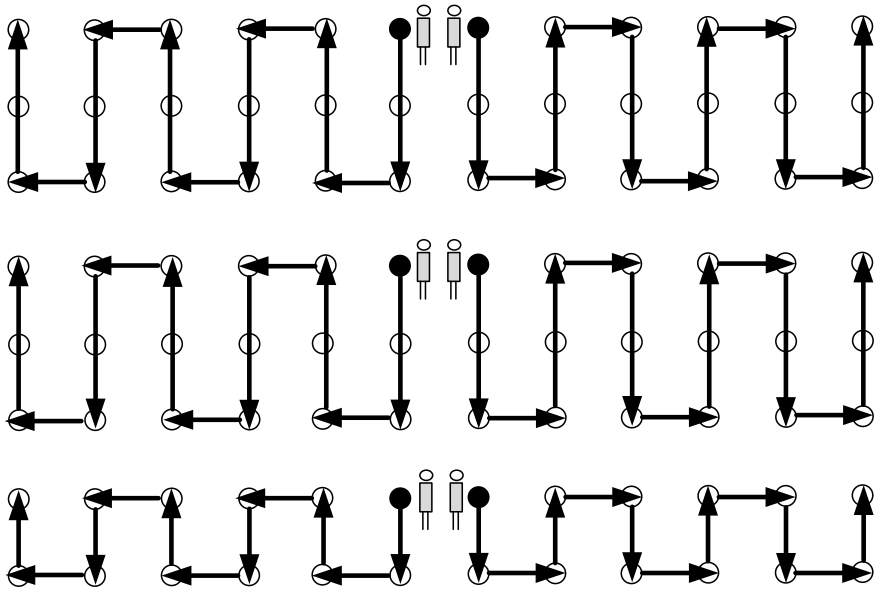

Figure 6.4: Torus Decontamination by Multiple Mobile Agents with Temporal Immunity: Strategy 1

Algorithm 3 Torus Decontamination with Temporal Immunity: Strategy 1

input: A contaminated $m \times n$-Torus ($m < n$) with temporary immunity t and A agents

output: The given torus is decontaminated

Variables:

$|A| = 2 \times \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil, i, j, k, P(i,j), MCount_k$

InitializationTorus

DecontaminationTorus

---

function InitializationTorus

All agents are in node $P(0,0)$

for $(k = 1, k \leq |\frac{A}{2}|, k++)$ **do**

for $(i = 0, i \leq m-1, i = i + \lceil \frac{t}{2} \rceil)$ **do**

Move a_k to the node $P(i,0)$

end for

end for

for $(k = 1, |\frac{A}{2}| \leq k < A+1, k++)$ **do**

for $(i = 0, i \leq m-1, i = i + \lceil \frac{t}{2} \rceil)$ **do**

Move a_k to the node $P(i, n-1)$

end for

end for

end function

```
function DecontamintionTorus
    for (j = 0, j < ⌈n/2⌉ AND k < A/2 + 1, j + 2) do
        for (k = 1, k ≤ A/2, K + +) do
            MCount_k = 0
            for (i = 0, i < ⌈t/2⌉, i + +) do
                Move(i + 1, j); MCount_k + +
            end for
            if (MCount = ⌈t/2⌉ − 1) then
                Move(i, j + 1)
            else
                Wait (⌈t/2⌉ − MCount_k); Move(i, j + 1)
            end if
            MCount_k = 0
            for (i = ⌈t/2⌉ − 1, i < 0, i − −) do
                Move(i − 1, j); MCount_k + +
            end for
            if (MCount < ⌈t/2⌉ − 1) then
                Move (i, j + 1)
            else
                Wait (⌈t/2⌉ − MCount_k); Move (i, j + 1)
            end if
        end for
    end for
    for (j = n − 1, j > ⌈n/2⌉ AND k > A/2, j − 2) do
        for (k = A/2 + 1, k < |A| + 1, K + +) do
            MCount_k = 0
            for (i = 0, i < ⌈t/2⌉, i + +) do
                Move(i + 1, j); MCount_k + +
            end for
            if (MCount = ⌈t/2⌉ − 1) then
                Move(i, j − 1)
            else
                Wait (⌈t/2⌉ − MCount_k); Move(i, j − 1)
            end if
            MCount_k = 0
            for (i = ⌈t/2⌉ − 1, i < 0, i − −) do
                Move(i − 1, j); MCount_k + +
            end for
            if (MCount < ⌈t/2⌉ − 1) then
                Move (i, j − 1)
            else
                Wait ((⌈t/2⌉ − MCount_k)); Move (i, j − 1)
            end if
        end for
    end for
end function
```

Theorem 6.7. *A torus with temporal immunity t can be decontaminated by $|A| = 2\lceil\frac{2m-1}{t}\rceil$ agents.*

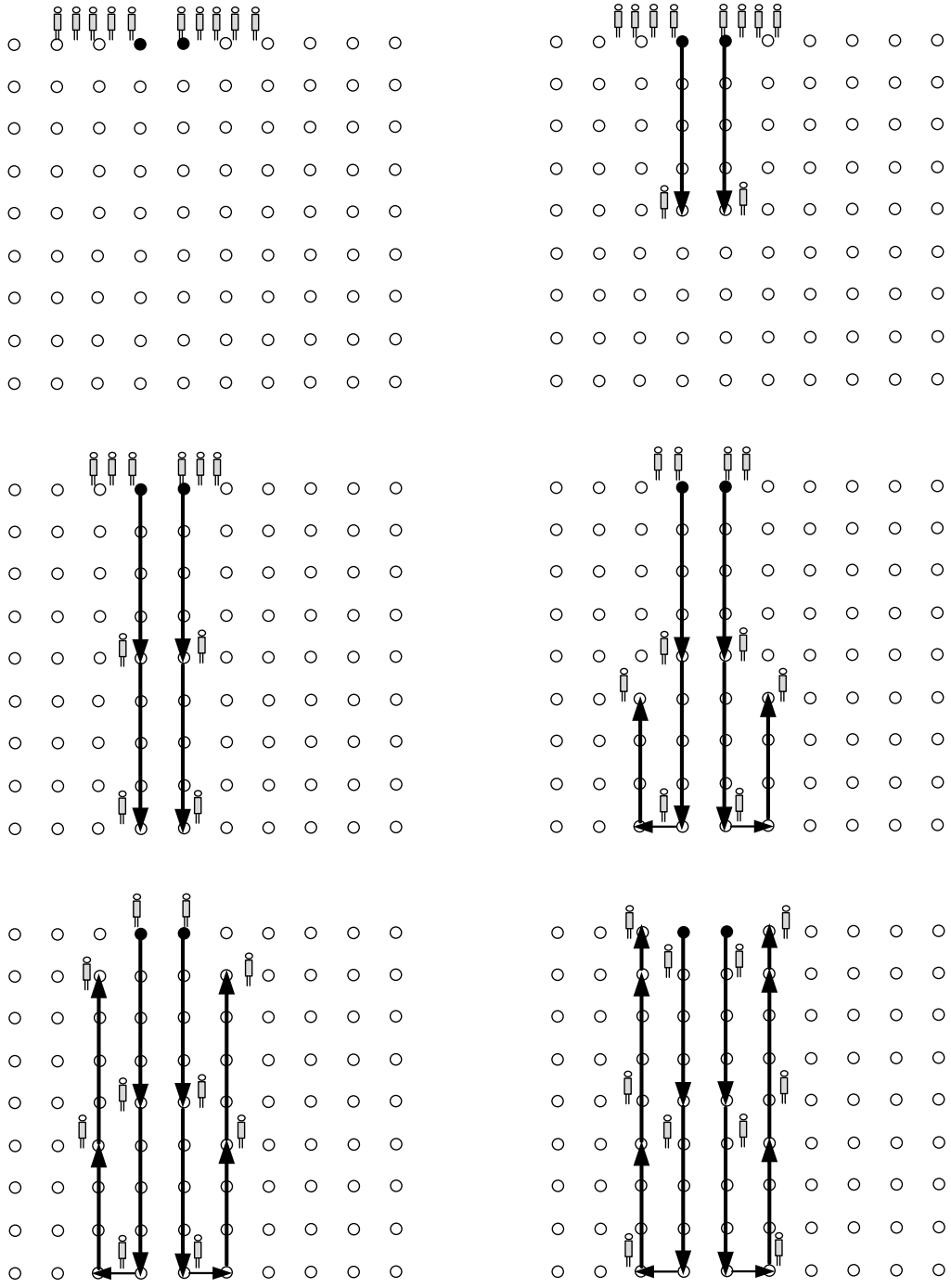

Figure 6.5: Torus Decontamination by Multiple Mobile Agents with Temporal Immunity: Strategy 2

Algorithm 4 Torus Decontamination with Temporal Immunity: Strategy 2

input: A contaminated $m \times n$-torus ($m < n$) with temporary immunity t and A agents
output: The given torus is decontaminated

Variables:
$|A| = 2 \times \lceil \frac{2m-1}{t} \rceil$, $i, j, P(i, j)$
$\frac{|A|}{2}$ agents are in $P(0, j)$, and $\frac{|A|}{2}$ are in $P(0, j+1)$
Decontamination:
for $(k = 0, k \leq \frac{|A|}{2}, k++)$ **do**
  Wait $(K \times t)$
  for $(x = j, x \leq j + \frac{n-2}{2}, x--)$ **do**
    for $(i = 0, i \leq n-2, i++)$ **do**
      Move to $P(i+1, x)$
    end for
    Move to $P(i, x-1)$
    for $(i = m-1, i \geq 1, i--)$ **do**
      Move to $P(i-1, x)$
    end for
    Move to $P(i, x-1)$
  end for
end for
for $(k = \frac{|A|}{2} + 1, k \leq |A|, k++)$ **do**
  Wait $((K - \frac{|A|}{2} + 1) \times t)$
  for $(x = j+1, x \leq j + 1 + \frac{n-2}{2}, x++)$ **do**
    for $(i = 0, i \leq n-2, i++)$ **do**
      Move to $P(i+1, x)$
    end for
    Move to $P(i, x+1)$
    for $(i = m-1, i \geq 1, i--)$ **do**
      Move to $P(i-1, x)$
    end for
    Move to $P(i, x+1)$
  end for
end for

Notice that with a single agent we cannot decontaminate the torus with the previous algorithm, but we can use a strategy similar to the strategy used for the case of mobile cellular automata (see Figure 6.6), then we can conjecture that:

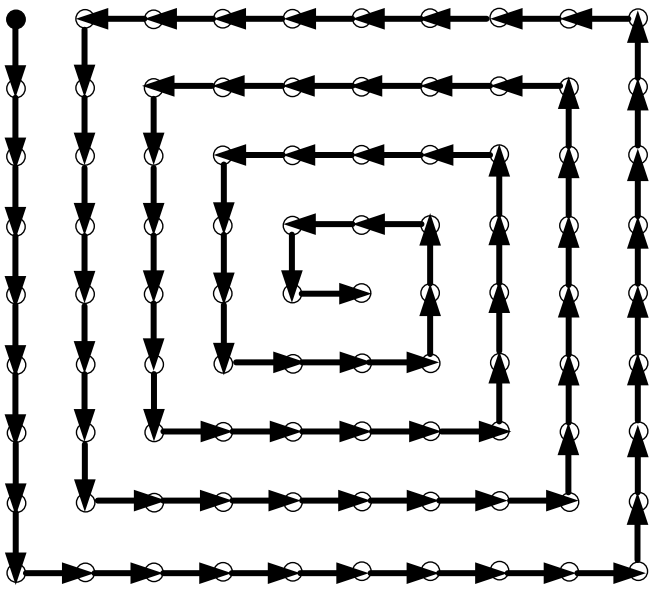

Figure 6.6: Torus Decontamination by Single Mobile Agent with Temporal Immunity

Conjecture 6.8. *A torus with temporal immunity* $t = 2(m-1) + 2(n-1) - 1$ *can be decontaminated by a single mobile agent.*

6.3 Conclusion

We have studied the problem of network decontamination with temporal immunity by mobile agents. We have considered mesh and torus topologies. Our objective was to minimize the number of agents used to decontaminate the network while ensuring that no recontamination can occur, as well as the solution strategy which is the algorithm enabling a minimal team of cleaners to perform the task. We proposed two solutions for the mesh decontamination with temporal immunity by mobile agents and we proved their correctness. In the case of a single agent, both strategies are optimal. In the case of torus, two strategies are described and their correctness is proven for the decontamination of a torus with multiple mobile agents k (where $k \geq 2$). In the case of a single agent, another strategy is described. Table 6.1 summarizes our results.

Topology	Number of Agents	Immunity Time
Mesh	$k = \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$	t
	$k = \lceil \frac{2m-1}{t} \rceil$	t
	$k = 1$	$t = 2m - 1^*$
Torus	$k = 2 \times \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$	t
	$k = 2 \times \lceil \frac{2m-1}{t} \rceil$	t
	$k = 1$	$t = 2(m-1) + 2(n-1) - 1^*$

Table 6.1: Chapter 6: Summary of Results

*The proposed strategy is an optimal solution

Chapter 7

Conclusion and Future Work

In this thesis we have studied the *decontamination* problem with and without temporal immunity, with three different models (mobile agents, cellular automata, and mobile cellular automata). In particular we have proposed different solutions and evaluated them according to the established complexity measure. Our goal was to better understand the impact that additional assumptions (memory, cloning, visibility, neighbourhood, locality) have on the complexity of our solutions. We have studied the notion of temporal immunity and analyzed its impact on the process of monotone decontamination in the mesh and torus topologies. Many problems and questions are opened by this thesis.

7.1 Summary

First in chapter 2, we provided a comprehensive survey of the history and current state of research in computation by mobile agents in safe and unsafe environments, and in cellular automata and Turmites. This survey helps in understanding in more depth the decontamination problem and different techniques useful in our study of network decontamination problems .

In chapter 3, we formally described the three models used in this work. We considered the mobile agent model, which is a model commonly employed in the distributed computing community, and where the decontamination is performed by mobile entities moving in the network. We then described the cellular automata model, where there is a cellular space where cells change their state synchronously according to local rules. Finally, we introduced the mobile cellular automata model, a combination of the two

previous models, where simple entities possessing visibility of their neighbourhood move synchronously on a cellular space following local rules.

In chapter 4, we considered the problem of decontaminating a two-dimensional three-states cellular automata with and without temporal immunity, where the spread of faults, as well as the decontamination process, are regulated by classical cellular automata local rules. We focused on finite and circular cellular automata with Von Neumann and Moore neighbourhoods. In each case, we described cellular automata rules to achieve decontamination. We measured the efficiency of our sets of rules by considering the number of simultaneous sites in the disinfecting state and, with this metric, we have shown that most rules are optimal.

In chapter 5, we focused on mobile cellular automata to look at the impact that active cells, in an otherwise reactive environment, have on the decontamination problem. We observed that, if cloning is not allowed, with the Von Neumann neighbourhood we obtain a more general solution than in the case of cellular automata with the same neighbourhood. The efficiency of the proposed solutions depends on the relationship between the number of cells in the network (n) and the time immunity (t) and is always better than the one devised for cellular automata. Note that the solutions for cellular automata could be also simulated by mobile cellular automata, but it would be efficient only for large immunity times ($t \geq n-2$). On the other hand, with the Moore neighbourhood the solution obtained with cellular automata is better than the one devised for mobile cellular automata, possibly because the lack of cloning capabilities of the agents is a bigger constraint than the lack of active cells. When the agents can clone, the Von Neumann neighbourhood allows to obtain the same results as the ones obtained for cellular automata with the Moore neighbourhood, thus showing that the extra power of the active cells is traded off with the enlarged neighbourhood.

In chapter 6, we studied the classical mobile agent model. This model was extensively studied in the context of decontamination. We presented two different algorithms for the mesh and torus topologies, and optimality in the case of a single agent in the mesh was proven. As with the solutions proposed in the literature, the main objective is to minimize the number of agents required to decontaminate the network. We also focus on understanding what impact the power of the agents and the underlying system has on the complexity of the problem. We noticed that the visibility capability is as powerful

as memory and communication using whiteboard capabilities together.

In chapter 7, we gave a summary of our results and showed directions for future work. The following two Tables 7.1, 7.2 summarize our results in this thesis.

Model	Neighbourhood	Active Entities	Time
Cellular Automata	Von Neumann	$k = n$	$t = 1^*$
		$K = 1, 2, 4$	$t = \frac{4}{k} \times (n-1) - 1$
	Moore	$k \geq 2$	$t \geq \lceil \frac{(2n-1)}{k} \rceil$
		$k = 1$	$t \geq (2n-1)^*$
Mobile Cellular Automata	Von Neumann	$k = n$	$t = 1^*$
		$k = n$ (with cloning)	$t = 1^*$
		$k = 1$	$t = 2n - 1^*$
		$k = \min\{\frac{2n}{t_1+1}, \frac{3n}{t_2+1}\}$	t
		$k = \lceil \frac{2n}{t} \rceil$(with cloning)	t
	Moore	$k = n$	$t = 1^*$
		$k = \frac{2n}{t+1}$ If $2n \mod (t+1) = 0$ $k = \times \lfloor \frac{2n}{t+1} \rfloor + 2$ Otherwise	t
Mobile Agents	NA†	$k = 1$	$t = 2m - 1^*$
		$k = \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$	t
		$k = \lceil \frac{2m-1}{t} \rceil$	t

Table 7.1: Summary of Results in the Mesh.

*The proposed strategy is an optimal solution

†Not Applicable in this model since agent does not have visibility (cannot see any neighbours)

Model	Neighbourhood	Active Entities	Time
Cellular Automata	Von Neumann	$k = 2n$	$t = 1^*$
	Moore	$k \geq 2$	$t = d_{max}{}^{\dagger}$
Mobile Cellular Automata	Von Neumann	$k = 2n$	$t = 1^*$
		$k = 2n$ (with cloning)	$t = 1^*$
		$k = \min\{2\frac{2n}{t_1+1}, 2\frac{3n}{t_2+1}\}$	t
		$k = 1$	$t = 4(n-1) - 1^*$
		$k = 2, 3$	$t = (n+1)$
		$k = 2 \times \lceil \frac{2n}{t} \rceil$ (with cloning)	t
	Moore	$k = 2 \times \frac{2n}{t+1}$ If $2n \mod (t+1) = 0$ $k = 2 \times \lfloor \frac{2n}{t+1} \rfloor + 3$ Otherwise	t
Mobile Agents	NA‡	$k = 2 \times \lceil \frac{m}{\lceil \frac{t}{2} \rceil} \rceil$	t
		$k = 2 \times \lceil \frac{2m-1}{t} \rceil$	t
		$k = 1$	$t = 2(m-1) + 2(n-1) - 1^*$

Table 7.2: Summary of Results in the Torus.

7.2 Future Work

In addition to what we have studied in this thesis, there are other works and open issues related to the network decontamination with temporal immunity problem. We have only covered a specific aspect in this research area. For future studies, we list some possible future work on the network decontamination with temporal immunity problem.

First and foremost, what happens in other classes of networks? Is the presence of temporal immunity going to result in improvements in other networks, comparable to the ones we observed in the case of mesh and torus? And if not, why not?

For the networks studied in this thesis, it is possible to improve some of the results, and/or prove their optimality. For example, in the case of mesh decontamination with temporal immunity by mobile agents, new forms of immunities such as temporary and

*The proposed strategy is an optimal solution

†Not Applicable in this model since agent does not have visibility (cannot see any neighbours)

$^{\ddagger}d_{max}$ the maximum distance between two consecutive *decontaminating* cells or between a *decontaminating* cell and a corner at time 0.

majority based immunity, and solutions where agents with the same or different capabilities can be deployed randomly in the network without any prior known configuration. Do these solutions in any way achieve decontamination? Considering the problem of decontaminating an arbitrary network by a single agent, we want to determine the minimum time required to perform this task. Is this an NP-hard problem?

For the neighbourhood, we have considered the Von Neumann neighbourhood with distance $r = 1$ and the Moore neighbourhood with distance $r = 1$. We have already noticed the impact of a larger neighbourhood on the existence of a solution and its complexity. For example, in the case of circular mobile cellular automata, what would the impact of the neighbourhood be if we consider the Von Neumann neighbourhood with distance $r \geq 2$.

We have assumed that the network and the agents are reliable. Our algorithms do not consider any broken network links, or failed network nodes and agents. Fault tolerance may be added in our algorithms in the future.

Another, more fundamental, question relates to the monotone nature of the solution protocols. If we remove monotonicity, what would be the minimum number of agents needed for decontamination of the network considered here? It is known that without temporal immunity the same number of agents are needed in some topology regardless of whether or not monotonicity is required [7].

We believe that investigations in these areas will provide useful insights in to network decontamination.

Bibliography

[1] L. Acerbi, A. Dennunzio, and E. Formenti. Shifting and Lifting of Cellular Automata. In *3rd Conference on Computability in Europe: Computation and Logic in the Real World (CIE)*, pages 1–10, 2007.

[2] L. Acerbi, A. Dennunzio, and E. Formenti. Conservation of some Dynamical Properties for Operations on Cellular Automata. *Theoretical Computer Science*, 410:3685–3693, 2009.

[3] S. Albers and M. R. Henzinger. Exploring Unknown Environments. In *29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 416–425, 1997.

[4] B. Awerbuch, M. Betke, R. L. Rivest, and M. Singh. Piecemeal Graph Exploration by a Mobile Robot. *Information and Computation*, 152(2):155–172, 1999.

[5] L. Barrière, P Flocchini, P. Fraigniaud, and N. Santoro. Capture of an Intruder by Mobile Agents. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 200–209, 2002.

[6] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Election and Rendezvous in Fully Anonymous Systems with Sense of Direction. In *10th Internaltional Colloquium on Structural Information Complexity (SIROCCO)*, pages 17–32, 2003.

[7] L. Barrière, P. Fraigniaud, N. Santoro, and D. M. Thilikos. Searching Is Not Jumping. In *29th International Workshop on Graph Theoretic Concepts in Computer Science (WG)*, pages 34–45, 2003.

[8] V. Baston and S. Gal. Rendezvous Search when Marks are Left at the Starting Points. *Naval Research Logistics*, 48(8):722–731, 2001.

[9] S. Belgacem and N. Fatès. Three Emergent Phenomena in the Multi-Turmite System and their Robustness to Asynchrony. Technical Report INRIA-00462438, INRIA, 2010.

[10] M. A. Bender, A. Fernández, D. Ron, A. Dennunzio Sahai, and S. P. Vadhan. The Power of a Pebble: Exploring and Mapping Directed Graphs. *Information and Computation*, 176(1):1–21, 2002.

[11] M.A. Bender and D.K. Slonim. The Power of Team Exploration: Two Robots can Learn Unlabeled Directed Graphs. In *35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 75–85, 1994.

[12] E. Berger. Dynamic Monopolies of Constant Size. *Journal of Combinatorial Theory Series B*, 83(2):191–200, 2001.

[13] O. Beuret and M. Tomassini. Behaviour of Multiple Generalized Langton's Ants. In *Artificial Life V Conference (ALIFE V)*, pages 45–50, 1997.

[14] L. Blin, P. Fraigniaud, N. Nisse, and S. Vial. Distributed Chasing of Network Intruders. *Theoretical Computer Science*, 399(1-2):12–37, 2008.

[15] J. P. Boon. How Fast does Langton's Ant Move ? *Journal of Statistical Physics*, 102(1-2):355–360, 2001.

[16] R. Breish. Intuitive Approach to Speleotopology. *Southwestern cavers*, 6(5):72–78, 1967.

[17] G. Cattaneo, A. Dennunzio, and L. Margara. Solution of some Conjectures about Topological Properties of Linear Cellular Automata. *Theoretical Computer Science*, 325:249–271, 2004.

[18] G. Cattaneo, M. Finelli, and L. Margara. Investigating Topological Chaos by Elementary Cellular Automata Dynamics. *Theoretical Computer Science*, 244(1-2):219–241, 2000.

[19] G. Cattaneo, E. Formenti, G. Manzini, and L. Margara. Ergodicity, Transitivity, and Regularity for Linear Cellular Automata over Z_m. *Theoretical Computer Science*, 233(1-2):147–164, 2000.

[20] V. Chevrier and N. Fatès. Multi-Agent Systems as Discrete Dynamical Systems: Influences and Reactions as a Modelling Principle. Technical Report INRIA-00345954, INRIA - LORIA, 2008.

[21] B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems*. Cambridge University Press, 1998.

[22] C. Cooper, R. Klasing, and T. Radzik. Searching for Black-Hole Faults in a Network Using Multiple Agents. In *International Conference On Principles Of DIstributed Systems (OPODIS)*, pages 320–332, 2006.

[23] J. Czyzowicz, A. Kosowski, and A. Pelc. How to Meet when you Forget: Log-Space Rendezvous in Arbitrary Graphs. In *29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 450–459, 2010.

[24] J. Czyzowicz, D. R. Kowalski, E. Markou, and A. Pelc. Complexity of Searching for a Black Hole. *Journal Fundamenta Informaticae*, 71(2-3):229–242, 2006.

[25] J. Czyzowicz, D. R. Kowalski, E. Markou, and A. Pelc. Searching for a Black Hole in Synchronous Tree Networks. *Combinatorics, Probability and Computing*, 16(4):595–619, 2007.

[26] Y. Daadaa, P. Flocchini, and N. Zaguia. Network Decontamination with Temporal Immunity by Cellular Automata. In *9th International Conference on Cellular Automata for Research and Industry (ACRI)*, pages 287–299, 2010.

[27] Y. Daadaa, P. Flocchini, and N. Zaguia. Decontamination with Temporal Immunity by Mobile Cellular Automata. In *International Conference on Scientific Computing (CSC)*, pages 172–178, 2011.

[28] M. D'amico, G. Manzini, and L. Margara. On Computing the Entropy of Cellular Automata. *Theoretical Computer Science*, 290(3):1629–1646, 2003.

[29] S. Das, P. Flocchini, A. Nayak, and N. Santoro. Distributed Exploration of an Unknown Graph. In *12th Internaltional Colloquium on Structural Information Complexity (SIROCCO)*, pages 99–114, 2005.

[30] X. Deng and C. H. Papadimitriou. Exploring an Unknown Graph (Extended Abstract). In *FOCS*, pages 355–361, 1990.

[31] A. Dennunzio, P .Di Lena, E .Formenti, and L. Margara. On the Directional Dynamics of Additive Cellular Automata. *Theoretical Computer Science*, 410:4823–4833, 2009.

[32] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic Rendezvous in Graphs. In *ESA*, pages 184–195, 2003.

[33] P. di Lena and L. Margara. Computational Complexity of Dynamical Systems: The Case of Cellular Automata. *Information and Computation*, 206(9-10):1104–1116, 2008.

[34] P. di Lena and L. Margara. On the Undecidability of the Limit Behavior of Cellular Automata. *Theoretical Computer Science*, 411(7-9):1075–1084, 2010.

[35] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree Exploration with Little Memory. *Journal of Algorithms*, 51(1):38–63, 2002.

[36] S. Dobrev, P. Flocchini, R. Kralovic, P. Ruzicka, G. Prencipe, and N. Santoro. Black Hole Search in Common Interconnection Networks. *Networks*, 47(2):61–71, 2006.

[37] S. Dobrev, P. Flocchini, R. Kralovic, and N. Santoro. Exploring an Unknown Graph to Locate a Black Hole Using Tokens. In *IFIP TCS*, pages 131–150, 2006.

[38] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Multiple Agents Rendezvous in a Ring in Spite of a Black Hole. In *International Conference On Principles Of DIstributed Systems (OPODIS)*, pages 34–46, 2003.

[39] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a Black Hole in Arbitrary Networks: Optimal Mobile Agents Protocols. *Distributed Computing*, 19(1):1–19, 2006.

[40] S. Dobrev, P. Flocchini, G Prencipe, and N. Santoro. Mobile Search for a Black Hole in an Anonymous Ring. *Algorithmica*, 48(1):67–90, 2007.

[41] S. Dobrev, P. Flocchini, and N. Santoro. Improved Bounds for Optimal Black Hole Search with a Network Map. pages 111–122, 2004.

[42] S. Dobrev, P. Flocchini, and N. Santoro. Cycling Through a Dangerous Network: A Simple Efficient Strategy for Black Hole Search. In *26th International Conference on Distributed Computing Systems (ICDCS)*, page 57, 2006.

[43] S. Dobrev, R. Kralovic, N. Santoro, and W. Shi. Black Hole Search in Asynchronous Rings Using Tokens. In *6th Conference on Algorithms and Complexity (CIAC)*, pages 139–150, 2006.

[44] S. Dobrev, N. Santoro, and W. Shi. Using Scattered Mobile Agents to Locate a Black Hole in an Un-Oriented Ring with Tokens. *International Journal of Foundation of Computer Science*, 19(6):1355–1372, 2008.

[45] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic Exploration as Graph Construction. *Robotics and Automation, IEEE Transactions on*, 7(6):859 – 865, 1991.

[46] P. Flocchini. Contamination and Decontamination in Majority-Based Systems. *Journal of Cellular Automata*, 4(3):183–200, 2009.

[47] P. Flocchini, M. J. Huang, and F. L. Luccio. Decontamination of Hypercubes by Mobile Agents. *Networks*, 52(3):167–178, 2008.

[48] P. Flocchini, M. Jun Huang, and F. L. Luccio. Decontaminating Chordal Rings and Tori using Mobile Agents. *International Journal of Foundation of Computer Science*, 18(3):547–563, 2007.

[49] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without Communicating: Ring Exploration by Asynchronous Oblivious Robots. In *International Conference On Principles Of DIstributed Systems (OPODIS)*, pages 105–118, 2007.

[50] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Remembering Without Memory: Tree Exploration by Asynchronous Oblivious Robots. *Theoretical Computer Science*, 411(14-15):1583–1598, 2010.

[51] P. Flocchini, D. Ilcinkas, and N. Santoro. Ping Pong in Dangerous Graphs: Optimal Black Hole Search with Pure Tokens. In *22nd International Symposium on Distributed Computing (DISC)*, pages 227–241, 2008.

[52] P. Flocchini, R. Královič, P. Ruzička, and N. Santoro. On Time Versus Size for Monotone Dynamic Monopolies. In *7th Colloquium on Structural Information and Communication Complexity*, pages 111–126, 2000.

[53] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. Multiple Mobile Agent Rendezvous in a Ring. In *Latin American Theoretical INformatics (LATIN)*, pages 599–608, 2004.

[54] P. Flocchini, E. Lodi, F. Luccio, L. Pagli, and N. Santoro. Dynamic Monopolies in Tori. *Discrete Applied Mathematics*, 137(2):197–212, 2004.

[55] P. Flocchini, F. L. Luccio, and L. Xiuli Song. Size Optimal Strategies for Capturing an Intruder in Mesh Networks. In *Communications in Computing*, pages 200–206, 2005.

[56] P. Flocchini, B. Mans, and N. Santoro. Tree Decontamination with Temporary Immunity. In *19th International Symposium on Algorithms and Computation (ISAAC)*, pages 330–341, 2008.

[57] P. Flocchini, A. Nayak, and A. Schulz. Cleaning an Arbitrary Regular Network with Mobile Agents. In *2nd International Conference Distributed Computing and Internet Technology (ICDCIT)*, pages 132–142, 2005.

[58] P. Flocchini and N. Santoro. Distributed Security Algorithms by Mobile Agents. In *8th International Conference on Distributed Computing and Networking (ICDCN)*, pages 1–14, 2006.

[59] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc. Collective Tree Exploration. *Networks*, 48(3):166–177, 2006.

[60] P. Fraigniaud and D. Ilcinkas. Digraphs Exploration with Little Memory. In *21st Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 246–257, 2004.

[61] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph Exploration by a Finite Automaton. *Theoretical Computer Science*, 345(2-3):331–344, 2005.

[62] P. Fraigniaud and N. Nisse. Connected Treewidth and Connected Graph Searching. In *7th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 479–490, 2006.

[63] P. Fraigniaud and A. Pelc. Delays Induce an Exponential Memory Gap for Rendezvous in Trees. In *Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*, pages 224–232, 2010.

[64] A. Gajardo and E. Goles. Dynamics of a Class of Ants on a One-Dimensional Lattice. *Theoretical Computer Science*, 322(2):267–283, 2004.

[65] A. Gajardo, E. Goles, and A. Moreira. Generalized Langton's Ant: Dynamical Behavior and Complexity. In *18th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 259–270, 2001.

[66] A. Gajardo, A. Moreira, and E. Goles. Complexity of Langton's Ant. *Discrete Applied Mathematics*, 117(1-3):41–50, 2002.

[67] D. Gale, J. Propp, S. Sutherland, and S. Troubetzkoy. Further Travels with my Ant. *Mathematical Entertainments column, Mathematical Intelligencer*, 17(48-56), 1995.

[68] L. Gasieniec, E. Kranakis, D. Krizanc, and X. Zhang. Optimal Memory Rendezvous of Anonymous Mobile Agents in a Unidirectional Ring. In *32nd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 282–292, 2006.

[69] Y. Ginosar and R. Holzman. The Majority Action on Infinite Graphs: Strings and Puppets. *Discrete Mathematics*, 215:59–71, 2000.

[70] P. Glaus. Locating a Black Hole without the Knowledge of Incoming Link. In *Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 128–138, 2009.

[71] E. Goles, F. Fogelman-Soulie, and D. Pellegrin. Decreasing Energy Functions as a Tool for Studying Threshold Networks. *Discrete Applied Mathematics*, 12:261–277, 1985.

[72] E. Goles and J. Olivos. Periodic Behaviour of Generalized Threshold Functions. *Discrete Mathematics*, 30:187–189, 1980.

[73] N. Imani, H. Sarbazi-Azad, and A. Y. Zomaya. Capturing an Intruder in Product Networks. *Journal of Parallel and Distributed Computing*, 67(9):1018–1028, 2007.

[74] J. Kari. Reversible Cellular Automata. In Clelia De Felice and Antonio Restivo, editors, *Developments in Language Theory*, volume 3572 of *Lecture Notes in Computer Science*, pages 57–68. 2005.

[75] J. Kari. Undecidable Properties on the Dynamics of Reversible One-Dimensional Cellular Automata. In *Journées Automates Cellulaires (JAC)*, pages 3–14, 2008.

[76] J. Kari, P. Vanier, and T. Zeume. Bounds on Non-Surjective Cellular Automata. In *34th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 439–450, 2009.

[77] L. M. Kirousis and C. H. Papadimitriou. Searching and Pebbling. *Theoretical Computer Science*, 47:205–218, 1986.

[78] R. Klasing, A. Kosowski, and A. Navarra. Taking Advantage of Symmetries: Gathering of many Asynchronous Oblivious Robots on a Ring. *Theoretical Computer Science*, 411(34-36):3235–3246, 2010.

[79] R. Klasing, E. Markou, and A. Pelc. Gathering Asynchronous Oblivious Mobile Robots in a Ring. *Theoretical Computer Science*, 390(1):27–39, 2008.

[80] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Approximation Bounds for Black Hole Search Problems. In *International Conference On Principles Of DIstributed Systems (OPODIS)*, pages 261–274, 2005.

[81] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Hardness and Approximation Results for Black Hole Search in Arbitrary Networks. *Theoretical Computer Science*, 384(2-3):201–221, 2007.

[82] A. Kosowski, A. Navarra, and M. C. Pinotti. Synchronization Helps Robots to Detect Black Holes in Directed Graphs. In *International Conference On Principles Of DIstributed Systems (OPODIS)*, pages 86–98, 2009.

[83] D. R. Kowalski and A. Malinowski. How to Meet in an Anonymous Network. In *13th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 44–58, 2006.

[84] E. Kranakis, D. Krizanc, and E. Marcou. *The Mobile Agent Rendezvous Problem in the Ring*. Morgan and Claypool Publishers, 1st edition, 2010.

[85] E. Kranakis, D. Krizanc, and E. Markou. Mobile Agent Rendezvous in a Synchronous Torus. In *Latin American Theoretical INformatics (LATIN)*, pages 653–664, 2006.

[86] E. Kranakis, N. Santoro, C. Sawchuk, and D. Krizanc. Mobile Agent Rendezvous in a Ring. In *23rd International Conference on Distributed Computing Systems (ICDCS)*, pages 592–599, 2003.

[87] P. Kurka. Languages, Equicontinuity and Attractors in Cellular Automata. *Ergodic Theory and Dynamical Systems*, 17:417–433, 1997.

[88] S. Kutten and D. Peleg. Fault-Local Distributed Mending. *Journal of Algorithms*, 30:263–273, 1999.

[89] S. Kutten and D. Peleg. Tight fault Locality. *SIAM Journal on Computing*, 30:247–268, 2000.

[90] C .G .Langton. Studying Artificial Life with Cellular Automata. *Phys. D*, 2(1-3):120–149, 1986.

[91] A. S. LaPaugh. Recontamination Does Not Help to Search a Graph. *Journal of the ACM*, 40(2):224–245, 1993.

[92] F. Luccio and L. Pagli. A General Approach to Toroidal Mesh Decontamination with Local Immunity. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–8, 2009.

[93] F. Luccio, L. Pagli, and N. Santoro. Network decontamination in Presence of Local Immunity. *International Journal of Foundation of Computer Science*, 18(3):457–474, 2007.

[94] F. L. Luccio. Intruder Capture in Sierpinski Graphs. In *4th International Conference on Fun with Algorithms (FUN)*, pages 249–261, 2007.

[95] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro. Asynchronous Deterministic Rendezvous in Graphs. *Theoretical Computer Science*, 355(3):315–326, 2006.

[96] N. Megiddo, S. Hakimi, M. Garey, D. Johnson, and C. Papadimitriou. The Complexity of Searching a Graph. *Journal of the ACM*, 35(1):18–44, 1988.

[97] G. Moran. On the Period-Two-Property of the Majority Operator in Infinite Graphs. *Transactions of the American Mathematical Society*, 347(5):1649–1667, 1995.

[98] P. Panaite and A. Pelc. Exploring Unknown Undirected Graphs. *Journal of Algorithms*, 33(2):281–295, 1999.

[99] T. D. Parsons. Pursuit-Evasion in a Graph. *Theory and Applications of Graphs*, pages 426–441, 1976.

[100] T. D. Parsons. The Search Number of a Connected Graph. In *9th Southeastern Conference on Combinatorics, Graph Theory and Computing, Utilitas Mathematica*, pages 549–554, 1978.

[101] D. Peleg. Graph Immunity Against Local Influence. Technical report cs96-11, Mathematics and Computer Science, Weizmann Institue of Science, 1996.

[102] D. Peleg. Size Bounds for Dynamic Monopolies. *Discrete Applied Mathematics*, 86:263–273, 1998.

[103] D. Peleg. Local Majorities, Coalitions and Monopolies in Graphs: a Review. *Theoretical Computer Science*, 282(2):231–257, 2002.

[104] S. Poljak. On an Application of Convexity to Discrete Systems. *Discrete Applied Mathematics*, 12:27–32, 1986.

[105] S. Poljak and M. Sura. On Periodical Behavior in Societies with Symmetric Influences. *Combinatorica*, 1:119–121, 1983.

[106] J. Qiu. Best Effort Decontamination of Networks. Master's thesis, University of Ottawa, 2007.

[107] M. Sablik. Directional Dynamics for Cellular Automata: a Sensitivity to Initial Condition Approach. *Theoretical Computer Science*, 400(1-3):1–18, 2008.

[108] A. Spicher, N. Fatès, and O. Simonin. From Reactive Multi-Agents Models to Cellular Automata - Illustration on a Diffusion-Limited Aggregation Mode. In *International Conference on Agents and Artificial Intelligence (ICAART)*, pages 422–429, 2009.

[109] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.