Multi-agent Cooperative Cleaning of Expanding Domains

Yaniv Altshuler, Vladimir Yanovski, Israel A Wagner and Alfred M Bruckstein

The International Journal of Robotics Research 2011 30: 1037 originally published online 31 August 2010
DOI: 10.1177/0278364910377245

The International Journal of
Robotics Research
30(8) 1037–1071

DOI: 10.1177/0278364910377245
ijr.sagepub.com

Multi-agent Cooperative Cleaning of Expanding Domains

Yaniv Altshuler[1,2], Vladimir Yanovski[1], Israel A Wagner[1,3], Alfred M Bruckstein[1]

Abstract

Several recent works considered multi-a(ge)nt robotics in static environments. In this work we examine ways of operating in dynamic environments, where changes take place independently of the agents' activity. The work focuses on a dynamic variant of the cooperative cleaners *problem, a problem that requires several simple agents to clean a connected region of "dirty" pixels in* $\mathbf{Z}^2$*. A number of simple agents move in this dirty region, each having the ability to "clean" the place it is located in. Their goal is to jointly clean the given dirty region. The dynamic variant of the problem involves a deterministic expansion of dirt in the environment, simulating spreading of contamination or fire. Theoretical lower bounds for the problem are presented, as well as various impossibility results. A cleaning protocol for the problem is presented, and a wealth of experimental results testing its performance in comparison to the lower bounds. Several analytic upper bounds for the proposed protocol are also presented, accompanied with appropriate experimental results.*

Keywords

Collaborative cleaning, collaborative search, decentralized robots, grid search, expanding domains

1. Introduction

1.1. Motivation

In nature, ants, bees or birds often cooperate to achieve common goals and exhibit amazing feats of swarming behavior and collaborative problem solving. It seems that these animals are "programmed" to interact locally in such a way that the desired global behavior will emerge even if some individuals of the colony die or fail to carry out their task for some other reasons. It is suggested to consider a similar approach to coordinate a group of robots without a central supervisor, by using only local interactions between the robots. When this decentralized approach is used much of the communication overhead (characteristic to centralized systems) is saved, the hardware of the robots can be fairly simple and better modularity is achieved. A properly designed system should be readily scalable, achieving reliability and robustness through redundancy.

1.2. Multi-agent Robotics and Swarm Robotics

Significant research effort has been invested during the last few years in design and simulation of multi-agent robotics and intelligent swarm systems (see e.g. Mastellone et al. (2008) and Wagner and Bruckstein (2001), and earlier works like Brooks (1990) and Sen et al. (1994)).

Such designs are often inspired by biology (see Klos and van Ahee (2008) for evolutionary algorithms, Arkin and Balch (1997) for behavior-based control models, Su et al. (2009) for flocking and dispersing models and Weitzenfeld (2008) for predator–prey approaches), by physics (Hagelbäck and Johansson 2008), by sociology (Trajkovski and Collins 2009) or by economics applications (Michael et al. 2008).

Swarm-based robotic systems can generally be defined as highly decentralized agent collectives, i.e. groups of extremely simple agents, with limited communication, computation and sensing abilities, designed to be deployed together in order to accomplish various tasks.

Tasks that have been of particular interest to researchers in recent years include synergetic mission planning (Visser et al. 2008), patrolling (Agmon et al. 2008), fault-tolerant cooperation (Work et al. 2008), network security (Rehak et al. 2008), swarm control (Connaughton et al. 2008),

[1] Technion, Technion City, Haifa, Israel
[2] Deutsche Telekom Laboratories, Ben Gurion University of the Negev, Beer Sheva, Israel
[3] IBM Haifa Research Lab, Carmel Mountains, Haifa, Israel

Corresponding author:
Yaniv Altshuler
Technion, Technion City, Haifa 32000, Israel
Email: yanival@cs.technion.ac.il

design of mission plans (Manisterski et al. 2008), role assignment (Zheng and Koenig 2008), multi-robot path planning (Sawhney et al. 2008), traffic control (Agogino and Tumer 2008), formation generation (Bhatt et al. 2009), formation keeping (Bendjilali et al. 2009), exploration and mapping (Dudek et al. 1991), target tracking (Harmatia and Skrzypczykb 2009) and distributed search, intruder detection and surveillance (Hollinger et al. 2009).

1.3. Analysis of Multi-agent Robotics Systems

Unfortunately, the mathematical/geometrical theory of such multi-agent systems is far from being satisfactory, as pointed out for example in Efraim and D.Peleg (2007) and Bonabeau et al. (1999). A short discussion regarding the conceptual framework of intelligent swarms (or *swarm intelligence systems*) and the motivation behind their use appears in Section 2.

An interesting approach for the analysis and implementation of decentralized multi-agent systems is the *population problem*, discussed for example in Angluin et al. (2008), in which a consensus among a decentralized group of n agents is produced in $O(n \log n)$ time (even in the presence of $O(\sqrt{n})$ Byzantine adversary agents).

Our interest is focused on developing mathematical tools necessary for the design and analysis of such systems. The tools we can employ vary and it is only rarely that existing mathematical results can be readily deployed.

For example, in Wagner and Bruckstein (1994), it was shown that a number of agents can arrange themselves equidistantly in a row via a sequence of "local" linear adjustments, the convergence of the configuration to the desired one being exponentially fast. A different way of cooperation between agents, inspired by the behavior of ant colonies, was described in Bruckstein (1993). There it was proved that a sequence of ants engaged in a deterministic chain pursuit will find the shortest (i.e. straight) path from the ant hill to the food source, using only local interactions. In Bruckstein et al. (1997), the behavior of a group of agents on $\mathbf{Z}^2$ was investigated, where each ant-like agent is pursuing his predecessor, according to a discrete biased-random-walk model of pursuit on the integer grid. The average paths of such a sequence of a(ge)nts engaged in a chain of probabilistic pursuit was shown to converge to the "straight line" between the origin and the destination, and this too happens exponentially fast.

1.4. Multi-agent Robotics in Dynamic Environments

The vast majority of the works mentioned above discuss challenges involving multi agents operating in static domains. Such models, however, are often too limited to capture "real-world" problems, which, in many cases, involve elements external to the agents' system, which may influence their environment activities and goals. Designing robotic agents that can operate in such environments presents a variety of different mathematical challenges.

The main difference between algorithms designed for static environments and algorithms designed to work in dynamic environments is the fact that the agents' knowledge base (either central or decentralized) becomes partially unreliable, due to the changes that take place in the environment. Task allocation, cellular decomposition, domain learning and other approaches often used by multi-agent systems all become impractical, at least to some extent. Instead, the algorithms must include strong domain-independent components, which should ensure that the agents generate a desired effect, regardless of the changing environment.

One example is the use of multi agents for distributed search. While many works discuss search after "idle targets", recent works considered dynamic targets, meaning targets which after being detected by the searching robots respond by performing various evasive maneuvers intended to prevent their interception. This dynamic robotics problem, dating back to World War II operations research (see e.g. Thorndike (1946) and Morse and Kimball (1951)), requires the agents to cope with changes in the environment they are operating in. The first documented example for search in dynamic domains discussed a planar search problem, considering the patrol of a corridor between parallel borders. This problem was solved in Koopman (1980) in order to determine optimal patrol strategies for aircraft searching for moving ships in a channel.

A similar problem was presented in Vincent and Rubin (2004), where a system consisting of a swarm of UAVs (unmanned air vehicles) was designed to search for one or more "smart targets" (representing for example enemy units, or alternatively a lost friendly unit which should be found and rescued). In this problem the objective of the UAVs is to find the targets in the shortest time possible. While the swarm comprises relatively simple UAVs, lacking prior knowledge of the initial positions of the targets, the targets are equipped with strong sensors, capable of telling the locations of the UAVs from very long distances. The search strategy suggested in Vincent and Rubin (2004) defines *flying patterns* for the UAVs to follow, designed for scanning the (rectangular) area in such a way that the targets cannot re-enter areas which were already scanned by the swarm, without being detected. This problem was further discussed in Altshuler et al. (2008), where an improved decentralized search strategy was discussed, demonstrating a nearly optimal completion time, compared to the theoretical optimum, achievable by any search algorithm.

1.5. Dynamic Cooperative Cleaners

Here we shall examine a problem in which the agents must work in a dynamic environment—where changes take place, that are independent and certainly not caused by

the agents' activity. The work is a continuation of the research presented in Wagner et al. (2008), discussing the *cooperative cleaners* problem—a problem assuming a regular grid of connected rooms (pixels), parts of which are "dirty", the "dirty" pixels forming a connected region of the grid. On this dirty grid region several agents move, each having the ability to "clean" the place (the "room", "tile", "pixel" or "square") it is located in. We analyze the dynamic variant of this problem (described in Section 3), in which a deterministic evolution of the environment in assumed, simulating a spreading contamination (or spreading fire). Once again, the goal of the agents is to clean the spreading contamination in as little time as possible. In the spirit of Braitenberg (1984), we consider simple robots with only a bounded amount of memory (i.e. *finite-state machines*). Similar works concerning multi-agent systems may be found in Polycarpou et al. (2001), Rekleitis et al. (2004), Batalin and Sukhatme (2002) and Butler et al. (2001).

1.6. Contribution

This work is one of the first in which the performance of a multi-agent group in dynamic environments is studied analytically. The main contribution of this work is the analysis of the problem, leading to lower bounds for the resources (time and number of agents) required for guaranteeing a successful cleaning of a given expanding region, using any cleaning protocol (presented as Theorems 1 and 2). The lower bounds are agnostic to the capabilities of the agents or the algorithm they employ. Furthermore, an impossibility result for the problem—namely, a constructive way of producing instances of the problem which are impossible to cope with—is presented in Theorem 3.

In addition, this work discusses the design of a local behavior rule for a decentralized group of myopic and memoryless agents that leads to their successful collaboration in completing their global task. A cleaning protocol defining the local behavior of the agents, based on the static variant of the problem that was analyzed in Wagner et al. (2008), is presented. The differences between the two cleaning algorithms are very minor, and are certainly not the motivation of this work, as the essence of the work stands in the fact that it can cope with *dynamic domains* and in the performance analysis of the decentralized swarm system for such domains. As can be seen in our analytic bounds, a dynamic environment has a dramatic influence on the performance of the agents.

The performance of the protocol in expanding domains is studied and analytic upper bounds for the time needed for a collaborative group of robotic agents in order to guarantee a completion of the task are derived (Theorems 4, 5 and 6).

1.7. Paper's Organization

This paper is organized as follows: a discussion concerning swarm intelligence models and motivation is presented in Section 2. In Section 3, the formal problem is defined along with the aims and assumptions involved. Section 4 discusses lower bounds for the problem (namely, for the cleaning time with respect to any cleaning protocol), followed by an impossibility result. The cleaning protocol is presented in Section 5. The protocol's performance on dynamic environments is analyzed, resulting in several upper bounds over its cleaning time, presented in Section 6. A comparison of the proposed protocol to existing state of the art results in the literature appears in Section 7. Various experiments are presented in Section 8. We conclude this paper with a short discussion, in Section 9. Appendix 9.8 provides a complete proof of one of the theorems required for the lower bounds.

2. Swarm Intelligence—Framework and Motivation

2.1. Motivation

There is a growing demand for robotic solutions to increasingly complex and varied challenges. In the course of time it was realized that often a single robot is not the best solution in some application domains. Instead, teams of robots are called upon to work in an intelligently coordinated fashion, often achieving efficiency and reliability via redundancy.

In Dias and Stentz (2001), a detailed description of multi-robot application domains is presented, demonstrating how multi-robot systems can be more effective than a single robot in many areas. However, when designing such systems it should be noticed that simply increasing the number of robots assigned to a task does not necessarily improve the system's performance—multiple robots must intelligently cooperate and avoid disturbing each other's activity to achieve efficiency.

There are several key advantages to the reliance on *intelligent swarm robotics*. First, such systems inherently enjoy the benefit of parallelism. In task-decomposable application domains, robot teams can accomplish a given task more quickly than a single robot, by dividing the task into sub-tasks and executing them concurrently. In certain cases, a single robot may simply be unable to accomplish the task on its own (e.g. to carry a large and heavy object).

Second, decentralized systems tend to be, by their very nature, much more robust than centralized systems composed of a single but very complex unit. Generally speaking, a team of robots may provide a more robust solution by introducing redundancy, and by eliminating any single point of failure. While considering the alternative of using a single sophisticated robot, we should note that even the most complex and reliable robot may suffer an unexpected malfunction, which will prevent it from completing its task. When using a multi-agent system, on the other hand, even if a large number of the agents stop working for some reason, the entire group will often still be able to complete its task, although perhaps in a longer time

period. For example, for exploring a hazardous unmapped region (such as a minefield or the surface of Mars), the benefit of redundancy and robustness offered by a multi-agent system is quite obvious, and it is in this context that Rodney Brooks wrote his well-known "Fast, cheap and out of control" report (Brooks and Flynn 1989).

Another advantage of the decentralized swarm approach is the ability of dynamically reallocating sub-tasks between the swarm's units, thus adapting to unexpected changes in the environment. Furthermore, since the system is decentralized, it can respond relatively quickly to such changes, due to the benefit of locality, meaning the ability to swiftly respond to changes without the need of notifying a hierarchical "chain of command". Note that as the swarm becomes larger, this advantage becomes increasingly important.

In addition to the ability of quick response to changes, the decentralized nature of such systems also improves their scalability. The scalability of multi-agent systems is derived from relying on the "emergence" of task completion by inherently low communication and computation protocols implemented by the agents. As the tasks assigned to multi-agent-based systems become increasingly complex, so does the importance of the high scalability of the systems.

Finally, by using heterogeneous swarms, even more efficient systems could be designed, relying on the utilization of different types of agents whose physical properties enable them to perform much more efficiently certain special tasks.

2.2. Simplicity of the Agents

A key principle in the notion of swarms, or multi-agent robotics, is the simplicity of the agents. The notion of "simplicity" here means that the agents should be *significantly simpler* than a "single sophisticated robotic system", which may be necessary for the same purpose. Hence, the capabilities and the resources of the agents are assumed to be limited in the following ways:

- Memory resources—basic agents are assumed to contain only $O(1)$ memory (i.e. the size of an agent's memory is independent of the size of the problem). This usually imposes many interesting limitations on the agents. For example, an agent can remember only a limited part of its history of its activities so far. Thus, protocols designed for agents with such limited memory resources are usually very simple and attempt to solve problems by relying on some (necessarily local) basic patterns arising in the environment.
- Sensing capabilities—the agents are considered capable of seeing or probing a limited portion of their environment. For example, for agents moving on a 100×100 grid, a reasonable sensing radius may be 3 or 4, but certainly not 40.
- Computational resources—although agents are assumed to employ only limited computational resources, a formal definition of this constraint is hard to provide. In general, most of the time polynomial algorithms may be used.
- Communication capabilities are very limited as well: the issue of communication in multi-agent systems has been extensively studied in recent years. Distinctions between implicit and explicit communication are usually made, in which implicit communication occurs as a side effect of other actions, or inference on the world (see, e.g., Pagello et al. (1999)), whereas explicit communication is a specific act intended solely to convey information to other robots of the team. Explicit communication can be performed in several ways, such as a short-range point to point communication, or global broadcast or by using some sort of distributed shared memory. Such memory is often treated as a *pheromone*, used to convey small amounts of information between the agents by leaving traces in the environment (Yanovski et al. 2001; Felner et al. 2006). This approach is inspired from the coordination and communication methods used by many social insects. In fact, studies of ants (e.g. Adler and Gordon (1992)) show that the pheromone-based search strategies used by them in foraging for food in unknown terrains tend to be very efficient. In the spirit of designing a system which uses as simple agents as possible, we should aspire to endow it with as little communication capabilities as possible. With respect to the taxonomy of multi agents discussed in Dudek et al. (1996), we would be interested in using agents of type *COM-NONE* or if necessary of type *COM-NEAR* with respect to their communication distances, and of types *BAND-MOTION*, *BAND-LOW* or even *BAND-NONE* (if possible) with respect to their communication bandwidth. Although a certain amount of implicit communication can hardly be avoided (due to the simple fact that by changing the environment, the agents are constantly generating some kind of implicit information), explicit communication should be strongly limited or avoided altogether, in order to fit our paradigm (note that in many works in this field, this is not the case, and communication, as well as memory, resources, are often assumed to be abundant, in order to create complex cooperative systems).

In summary, we assume that the agents we design to be myopic, mute, senile and rather stupid, and we aim to program them with a protocol of local behavioral response to patterns in their immediate neighborhood.

3. The Dynamic Cooperative Cleaners Problem

The definition of the dynamic cooperative cleaners problem is very similar to its static variant (that can be found in Wagner et al. (2008)), albeit with several extensions. For the sake of readability, we list below a complete self-contained definition of the problem.

We assume that the time is discrete.

Definition 1. Let the undirected planar graph $G(V,E)$ have for V a two-dimensional integer grid $\mathbf{Z}^2$, whose vertices (or "*tiles*") have a binary property called "*contamination*". Let $cont_t(v)$ denote the contamination state of the tile v at time t, taking the value either "*on*" or 1 (for "dirty" or "contaminated") or "*off*" or 0 (for "clean" or "uncontaminated").

For two vertices $v,u \in V$, the edge (v,u) may belong to E at time t only if both of the following hold:

- v and u are 4*Neighbors* in $\mathbf{Z}^2$.
- $cont_t(v) = cont_t(u) = on$.

This however is a necessary but not a sufficient condition.

The edges of G, E represent the connectivity of the contaminated region. At $t = 0$ all the contaminated tiles are connected, namely

$$(v,u) \in E_0 \Updownarrow (v,u \text{ are 4}Neighbors \text{ in } \mathbf{Z}^2) \wedge (cont_0(v) = cont_0(u) = on).$$

Edges may be added to E only as a result of a contamination spread and can be removed only while contaminated tiles are cleaned by the agents (see below).

Definition 2. Let $F_t = (V_{F_t}, E_t)$ be the contaminated sub-graph of G at time t, i.e.

$$V_{F_t} = \{v \in \mathbf{Z}^2 \mid cont_t(v) = on\}.$$

We assume that F_0 is a single simply connected component, and the actions of the agents will be so designed that this property will be preserved.

Let a group of k agents that can move on the grid F_t (moving from a tile to its neighbor in one time step) be placed at time t_0 on F_0, at point $p_0 \in V_{F_t}$.

Each agent is equipped with a sensor capable of telling the *contamination* status of all tiles in the digital sphere of diameter 7, which surrounds the agent. An agent is also aware of other agents which are located in these tiles, and all the agents agree on a common correlation system. Each tile may contain any number of agents simultaneously. This information will later be required by the agents' cleaning protocol. Each agent is equipped with a memory of size $O(1)$ bits with respect to the size of the region[1].

The agents are indistinguishable. Namely, they can be counted, but they do not contain any unique ID.

When an agent moves to a tile v, it has the possibility of cleaning this tile (i.e. causing $cont(v)$ to become *off*. Once an agent cleaned a tile v, all the edges touching v are removed, namely

$$E_{t+1} = E_t \setminus \{(v,u) \mid (v,u) \in E_t \wedge cont(v) = off\}.$$

The agents do not have any prior knowledge of the shape or size of the sub-graph F_0 except that it is a single and simply connected component.

3.1. Spreading Contamination

Unlike the static variant of the problem, which was studied among others in Wagner et al. (2008), in this paper we are concerned with a dynamic contamination that spreads over time and, specifically, every d time steps. This is formally defined as follows.

Definition 3. Let d denote the number of time steps between contamination spreads. That is, if $t = nd$ for some positive integer n, then

$$\forall v \in F_t,\ \forall u \in 4Neighbors(v),\ cont_{t+1}(u) = on.$$

The evolution of the edges of E that represent connections between contaminated tiles can be defined in various ways. The simplest of which is merely stating that every couple of adjacent contaminated tiles is connected, namely we have the following definition.

Definition 4. At any given time step t,

$$\forall v \in V_t,\ \forall u \in V_t,\ (u \neq v) \rightarrow ((u,v) \in E_t).$$

An alternative definition of the evolution of E_t is discussed in Section 5.4, simulating an elastic "barrier" which preserves the simple connectivity of the contaminated region. This is later used in order to allow myopic and memoryless decentralized agents to collaboratively clean the spreading contaminated region, based on local interactions and information.

3.2. Agents' Goal

As is the case of the static variant of the cooperative cleaners problem, it is the agents' goal to clean F by eliminating the contamination entirely, meaning that the agents must ensure that

$$\exists t_{\text{success}} \text{ s.t. } F_{t_{\text{success}}} = \emptyset.$$

In addition, it is desired that the time t_{success} will be minimal.

In this work we impose the restriction of no central control and full "decentralization", i.e. all agents are identical and no explicit communication between the agents is allowed. An important advantage of this approach, in addition to the simplicity of the agents, is fault tolerance—even if almost all the agents cease to work before completion, the remaining ones will eventually carry on the mission, and complete it, if possible.

4. A Protocol-independent Lower Bound for Cleaning Time

4.1. Overview of the Results

Due to the dynamic nature of the problem, the shape of the contaminated region can change dramatically during the cleaning process. Therefore, since we know no easy way to

decide whether k agents can successfully clean an instance of the *dynamic cooperative cleaners* problem, producing bounds for the efficiency of any proposed cleaning protocol is important for estimating its efficiency.

Several analytic results are presented in this section. Given a contaminated shape F_0 with an initial area S_0, and k agents employing *any* cleaning protocol, two lower bounds for S_t (the area of F at time t) are derived and summarized in Theorems 1 and 2. Using the above, impossibility results are presented, as we can provide an algorithm to generate contaminated regions which are impossible to clean, given any pair of k and d.

4.2. Analysis

Definition 5. Let S_t denote the size of the contaminated region F at time t, namely the number of grid points (or tiles) in F_t. Actually, F defines a dichotomy of $\mathbf{Z}^2$ into F and $\overline{F} = \mathbf{Z}^2 \setminus F$.

Let us now discuss a general lower bound for the problem of dynamic cleaning. It is applicable for any cleaning protocol which might be used by the agents. This is achieved by showing that at a specific time t, $S_t > 0$, namely, the mission could not be completed until that time (regardless of the cleaning protocol used). Note that the completion of the cleaning mission at time t means that $S_t = 0$.

Theorem 1. *Using any cleaning protocol, the area of the contaminated region at time t can be recursively lower bounded, as follows:*

$$S_{t+d} \geq S_t - d \cdot k + \left\lfloor 2\sqrt{2 \cdot (S_t - d \cdot k) - 1} \right\rfloor.$$

Proof. Note that a lower bound for the cleaning time can be obtained by assuming that the agents are working with maximal efficiency, meaning that each at time step every agent cleans exactly one tile. After $d-1$ time steps k agents will therefore have cleaned $k \cdot (d-1)$ tiles, and thus we know that

$$S_{d-1} \geq S_0 - (d-1) \cdot k.$$

In the dth time step, the agents clean another portion of k tiles, but the remaining contaminated tiles spread their contamination to their 4*Neighbors* and cause new tiles to become contaminated. We are interested in the *minimal* number of tiles which can become contaminated at this stage. The minimal number of 4*Neighbors* of any number of tiles is achieved when the tiles are organized in the shape of a "digital sphere" (see a complete proof in Altshuler et al. (2006b) or Appendix 9.8, and a full discussion of isoperimetric inequalities for discrete grids and optimally packed shapes in Vainsencher and Bruckstein (2008)), as demonstrated in Figure 1. Therefore, for a region of any given area S the minimal number of its 4*Neighbors* is at least $2\sqrt{2 \cdot S_t - 1}$. As the number of tiles must be an integer value, we use $\left\lfloor 2\sqrt{2 \cdot S_t - 1} \right\rfloor$ to remain on the safe side.

Fig. 1. The left-hand chart shows an example of a "sphere" in the grid. Notice that this sphere has only 16 tiles in its 4*Neighbors* while the right-hand chart, containing a shape of the same area, has 20 tiles in its 4*Neighbors*.

After the dth time step we thus have the following:

$$S_d \geq S_0 - d \cdot k + \left\lfloor 2\sqrt{2 \cdot (S_0 - d \cdot k) - 1} \right\rfloor$$

and the conclusion follows. □

A graphical illustration of Theorem 1 is presented is Figure 2.

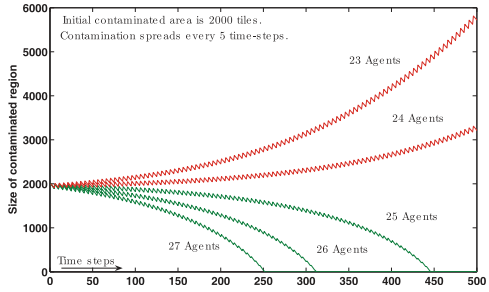

Fig. 2. Theorem 1 illustrated. The graph depicts a lower bound over the size of the contaminated region as a function of t, given various numbers of cleaning agents (while the initial area of the contaminated region is $S_0 = 2000$ and the contamination spreading latency is $d = 5$). The lower bound over the cleaning time can be interpreted as stating that the time for completion of the cleaning job will be higher than the point at which the graph of S_t hits 0. For example, observe how a complete cleaning is shown to be impossible for any swarm with 24 agents or less. Naturally, as Theorem 1 only takes into account the initial size of the contaminated region, different regions' size evolutions are expected to be seen for regions of various shapes of the same initial size—see such examples in Section 8. Note that as this bound is generic, and hence independent of the cleaning protocol used by the agents, the actual cleaning protocol used will have a significant effect on the actual cleaning efficiency.

4.3. Explicit Expression for the Lower Bound.

The following result is a direct lower bound for the size of the contaminated region, based on the recursive bound of Theorem 1.

Theorem 2. *Using any cleaning protocol, a lower bound for the area of the contaminated region at time $t = i \cdot d$ for some $i \in \mathbb{N}$ is the minimal positive S_t which is a solution of*

the following equation:

$$\mathcal{S}_t - \mathcal{S}_0 + \ln\left(\frac{\mathcal{S}_t - \frac{dk}{2}}{\mathcal{S}_0 - \frac{dk}{2}}\right)^{\frac{dk}{2}} = 2i,$$

where

$$\mathcal{S}_t \triangleq \sqrt{2(S_t - dk) - 1}, \quad \mathcal{S}_0 \triangleq \sqrt{2(S_0 - dk) - 1}.$$

Proof. Observe that by denoting $y_i \triangleq S_{id}$, the recursion of Theorem 1 can be written as

$$y_{i+1} - y_i \geq \left\lfloor 2\sqrt{2 \cdot (y_i - d \cdot k) - 1} \right\rfloor - d \cdot k.$$

Searching for the minimal area, we can look at the equation

$$y_{i+1} - y_i = \left\lfloor 2\sqrt{2 \cdot (y_i - d \cdot k) - 1} \right\rfloor - d \cdot k.$$

By dividing both sides by $\Delta i = 1$, we obtain

$$y_{i+1} - y_i \triangleq y' = \left\lfloor \sqrt{8y - 8\left(d \cdot k + \frac{1}{2}\right)} \right\rfloor - d \cdot k. \quad (1)$$

Note that the values of y' might be positive (stating an increase in the area), negative (stating a decrease in the area) or complex numbers (stating that the area is smaller than $d \cdot k$, and will therefore be cleaned before the next spread).

Let us denote $x^2 \triangleq 8y - 8\left(d \cdot k + \frac{1}{2}\right)$. After calculating the derivative of both sides of this expression, we see that

$$2x \cdot x' = 8y'$$

and, after using the definition of y' of Equation (1), we see that

$$2x \cdot \frac{dx}{di} = 2x \cdot x' = 8\left\lfloor \sqrt{8y - 8\left(d \cdot k + \frac{1}{2}\right)} \right\rfloor - 8d \cdot k$$

and, subsequently,

$$2x \cdot \frac{dx}{di} \leq 8\left(x - d \cdot k\right). \quad (2)$$

From Equation (2), a definition of di can be extracted:

$$di \geq \frac{1}{8} \cdot \frac{2x}{x - d \cdot k} dx \geq \frac{1}{8} \cdot \frac{2x - 2d \cdot k + 2d \cdot k}{x - d \cdot k} dx$$

$$\geq \frac{1}{4}\left(1 + \frac{d \cdot k}{x - d \cdot k}\right) dx.$$

The value of x can be achieved by integrating the previous expression as follows (note that we are interested in the equality of the two expressions):

$$\int_{i_0}^{i} di = \int_{x_0}^{x} \frac{1}{4}\left(1 + \frac{d \cdot k}{x - d \cdot k}\right) dx.$$

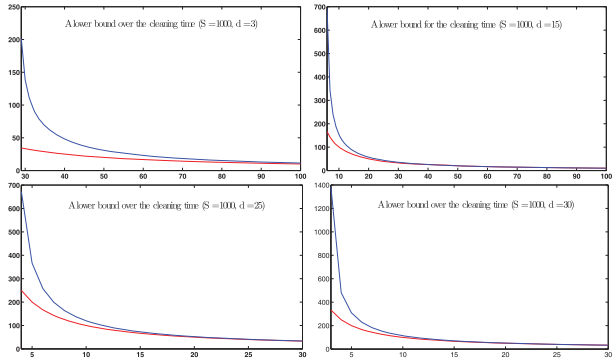

Fig. 3. Theorem 2: these graphs depict lower bounds for the cleaning time of several contaminated regions (the Y axis), as a function of the number of agents (the X axis, starting from the lowest number of agents which produces a successful cleaning). A comparison to the naive $f(k) = \frac{S_0}{k}$ function, which holds for static environments, is also included (the lower curve, marked in red). Notice again that this bound is generic, and therefore will hold for any cleaning protocol used.

After the integration, we can see that

$$i\Big|_{i_0}^{i} = \frac{1}{4}\left(x + d \cdot k \ln\left(x - d \cdot k\right)\right)\Big|_{x_0}^{x}$$

and, after assigning $i_0 = 0$,

$$4i = x - x_0 + d \cdot k \ln \frac{x - d \cdot k}{x_0 - d \cdot k}.$$

Rewriting this in terms of y, we get

$$2i = \sqrt{2(y - d \cdot k) - 1} - \sqrt{2(y_0 - d \cdot k) - 1}$$

$$+ \ln\left(\frac{\sqrt{2(y - d \cdot k) - 1} - \frac{d \cdot k}{2}}{\sqrt{2(y_0 - d \cdot k) - 1} - \frac{d \cdot k}{2}}\right)^{\frac{d \cdot k}{2}}$$

and, returning to the original size variable S_t, we see that

$$2i = \sqrt{2(S_{id} - dk) - 1} - \sqrt{2(S_0 - dk) - 1}$$

$$+ \ln\left(\frac{\sqrt{2(S_{id} - dk) - 1} - \frac{dk}{2}}{\sqrt{2(S_0 - dk) - 1} - \frac{dk}{2}}\right)^{\frac{dk}{2}}$$

and the rest follows. □

Theorem 2 provides an easy way of calculating the minimal possible time it will take a given number of agents to clean some portion of any given contaminated region (and, specifically, to clean it entirely). A lower bound for the cleaning time of k agents for various contaminated regions based on this theorem appears in Figures 3, 4 and 5. A lower bound for the minimal number of agents required to guarantee a successful cleaning of a contaminated region, as a function of the contamination spreading speeds, is shown in Figure 6, for several values of the initial contaminated

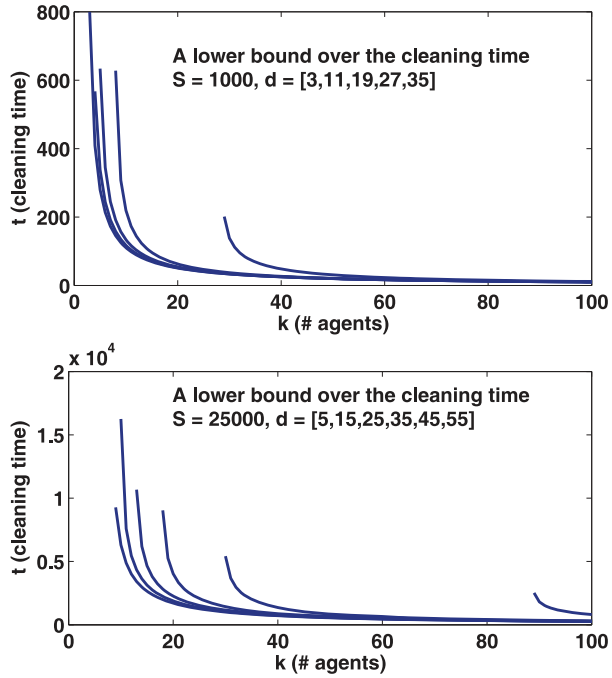

Fig. 4. Theorem 2: these graphs depict lower bounds for the cleaning time of several contaminated regions (the Y axis), as a function of the number of agents (the X axis), for various values of d. Notice again that this bound is generic, and therefore will hold for any cleaning protocol used.

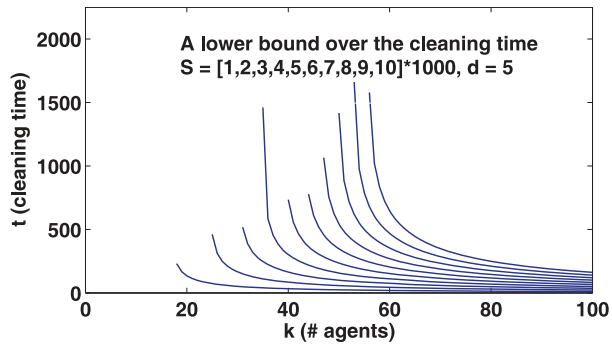

Fig. 5. Theorem 2: these graphs depict lower bounds for the cleaning time of several contaminated regions (the Y axis), as a function of the number of agents (the X axis), for various values of the initial area S_0.

area. The influence of the initial contaminated area over the minimal number of agents required to guarantee a complete cleaning is demonstrated in Figure 7, in which the ratio $\frac{K_{\min}(S_0,d)}{K_{\min}(\frac{1}{10}S_0,d)}$ (as a function of d) is presented (suggesting that when $K_{\min}(n,d)$ denotes the minimal number of agents required to enable a cleaning of a contaminated region of an initial contaminated region of n tiles, for some contamination spreading d, then $K_{\min}(\alpha \cdot n,d) \approx \sqrt{\alpha} \cdot K_{\min}(n,d)$).

Another interesting illustration of Theorem 2 is presented in Figure 8, where the ratio $R_{S_0}(\alpha) \triangleq \frac{T_{\min}(\alpha \cdot S_0,D)}{T_{\min}(S_0,D)}$ (where $T_{min}(n,D)$ denotes the average lower bound for the cleaning time of $K_{\min}(n,d)$ agents, where $d = [2,3,4,\ldots,D]$) is calculated for various values of S_0 (and for $D = 20$). Based on these examples, it can be conjectured that $R_{S_0}(\alpha) \approx 2\sqrt{\alpha}$ regardless of S_0 and D.

4.4. Impossibility Result

While designing a system intended to use cleaning agents in order to guarantee a successful completion of the cooperative cleaners problem, various cleaning methods can be used. This, combined with the dramatic changes the initial geometric properties of the contaminated region can undergo, yields a great uncertainty as to the cleaning time of a group of agents using any protocol (and, for that matter, as to the possibility of guaranteeing a successful job completion, to begin with). While the theoretical lower bound shown above as well as the upper bounds which are presented in the next section can assist in decreasing this uncertainty, one might alternatively be concerned with the opposite question, namely, how can we guarantee that a group of k agents *will not* be able to clean a contaminated region entirely (regardless of the cleaning protocol being used, or the agents' sensors or communication capabilities). A solution to this problem is the following theorem.

Theorem 3. *Using any cleaning protocol, k agents cleaning a contaminated region of initial size S_0, which spreads every d time steps, will not be able to clean it if*

$$S_0 > \left\lceil \frac{1}{8}d^2k^2 + dk + \frac{1}{2} \right\rceil.$$

Proof. We require that the size of the region increases between each two spreads (thus generating an ever expanding region, impossible to clean) or, in other words, we require that

$$S_d - S_0 > 0.$$

Using Theorem 1, we know that regardless of the specific protocol used,

$$S_d \geq S_0 - d \cdot k + 2\sqrt{2 \cdot (S_0 - d \cdot k) - 1}$$

and therefore

$$2\sqrt{2 \cdot (S_0 - d \cdot k) - 1} > d \cdot k.$$

Some algebra then yields

$$S_0 > \left\lceil \frac{1}{8}d^2k^2 + dk + \frac{1}{2} \right\rceil.$$

Theorem 3 is illustrated in Figure 9. □

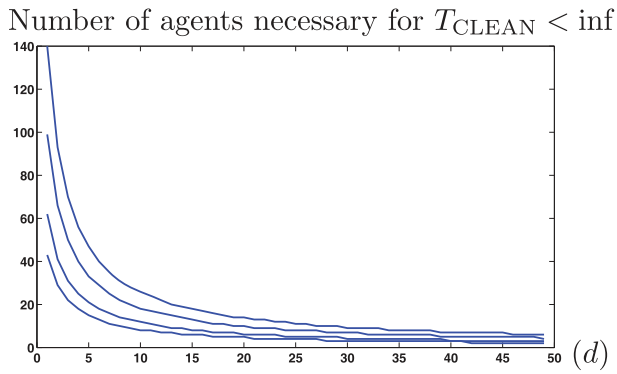

Fig. 6. Theorem 2: this graph represents lower bounds for the minimal number of agents required to guarantee a successful cleaning of several contaminated regions, as a function of the contamination spreading speed d. The lower curve depicts the minimal number of agents for an initial region of $S_0 = 1000$, for spreading speeds between 2 and 50. Similarly, the other curves represent a similar function for $S_0 = 2000, 5000$ and 10000, respectively.

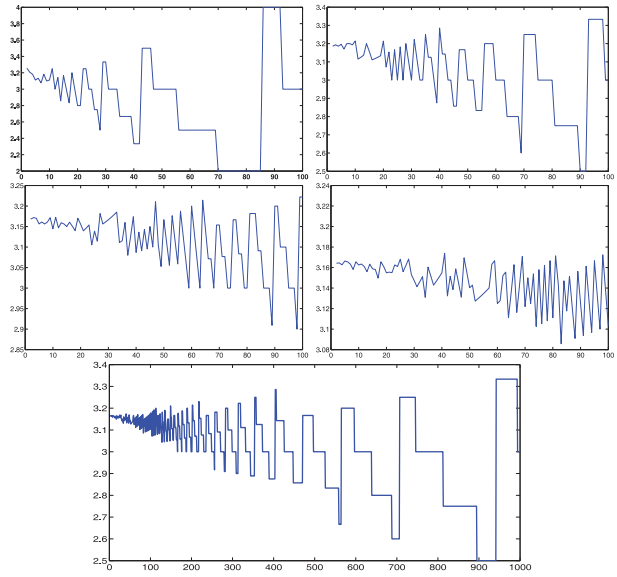

Fig. 7. An example of Theorem 2. These graphs demonstrate the influence of the change in the area of the initial contaminated region over the change in the lower bound for the minimal number of agents required to guarantee a successful cleaning of the region, namely, the ratio $\frac{k(S_0,d)}{k(\frac{1}{10}S_0,d)}$ (as a function of d), when $k(S,d)$ represents the lower bound over the minimal number of agents, for a region of area S and a spreading speed d. The values of S_0 of the four upper graphs are 10000, 100000, 1000000 and 10000000, while the values of d are between 2 and 100. A continuation of the last graph (i.e. in which $S_0 = 10000000$) appears in the lower graph, in which $2 \leq d \leq 1000$. Observing these graphs it can be seen that, as expected, an increase in d results in a general decrease in the agents' ratio (disregarding the oscillatory nature of the function). The reason is that when d is increased, the problem becomes "easier", and therefore the number of agents required to complete the cleaning is smaller (regardless of the cleaning time itself). As to the effect of the initial area of the region, the ratio between the agents required for cleaning an area of S_0 tiles and the agents required for cleaning an area of $\frac{1}{10}S_0$ tiles is generally constant for smaller values of d (and surrounds $\sqrt{10}$), regardless of the ratio between S_0 and $\frac{1}{10}$.

5. The Cleaning Protocol

5.1. Overview of the Protocol

In order to solve the *dynamic cooperative cleaners* problem we shall argue that **CLEAN**—a cleaning protocol that was suggested in Wagner et al. (2008)—can be used for the dynamic variant of the problem as well.

Generalizing an idea from computer graphics (presented in Henrich (1994)), this protocol preserves the connectivity of the *contaminated* region by preventing the agents from cleaning *critical points*—points which when cleaned disconnect the contaminated region. This ensures that the agents stop only upon completing their mission. At each time step, each agent cleans its current location (assuming that it is not a critical point), and moves according to a local movement rule, creating the effect of a clockwise "sweeping" traversal along the boundary of the contaminated region. As a result, the agents "peel" layers from the region, while preserving its connectivity, until the region is cleaned entirely. An illustration of two agents working according to the protocol can be seen in Figure 10.

There are several implications to the fact that the contamination is now assumed to be expanding over time. First, the analysis of the protocol's performance becomes increasingly more complicated (moreover, the protocol's correctness itself must be redemonstrated). Second, unlike in the static case, a group of k agents can no longer be assumed to eventually clean a contaminated region given enough time (as shown for example in Theorem 3). Rather, the group of agents must be large enough, in order to guarantee a successful completion of the task (see more details in Section 6). The agents acting according to the **CLEAN** protocol face a problem which did not appear when the contaminated region was assumed to be

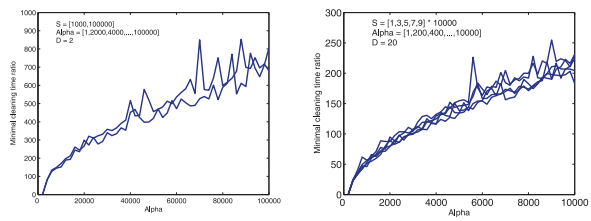

Fig. 8. An example of Theorem 2, illustrating the influence of the change in the contaminated region's initial size over the lower bound of the cleaning time of the smallest group of agents that is capable of cleaning it. Namely, the graphs demonstrate $R_{S_0}(\alpha) \triangleq \frac{T_{\min}(\alpha \cdot S_0, D)}{T_{\min}(S_0, D)}$ as a function of α. Notice that the value of $R_{S_0}(\alpha)$ is agnostic to the selection of the initial S_0 or of D.

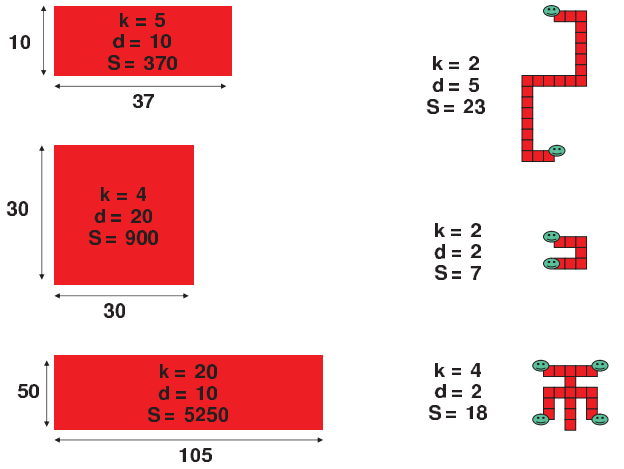

Fig. 9. An example of Theorem 3, presenting several examples for initial regions, spreading latencies and number of agents, which are guaranteed to be impossible for cleaning.

static, namely, the spread of the contamination, and, upon contamination spread, agents might find themselves located in tiles which are no longer boundary tiles. Once this happens, an agent will have to move towards the boundary tile which will enable the continuation of his planned traversal along the region.

For the sake of readability, we show here the complete description of the cleaning protocol which appears in Wagner et al. (2008). In the dynamic case, this cleaning protocol is sometimes called the **SWEEP** cleaning protocol and we shall use this name from now on.

5.2. Agents' Limitations and the Use of Local Rules

To the basic description of the protocol given above, there are several exceptions. As the agents are equipped with very limited sensing and storage capabilities, the basic structure of the protocol must be enhanced with a set of local rules, designed to induce a pseudo-synchronization between the agents. Those rules generate *resting* and *waiting* commands for the agents, capable of delaying their actions, either within the time step (until certain agents complete their cleaning process for this time step), or causing them to pause for a single time step, resuming their operation at the next time step only. A detailed description concerning the need for these additions appears in Sections 5.9 and 5.10. Note however that the rules in charge of the *resting* and *waiting* still obey the basic paradigm of this work, namely, they are local, use no prior knowledge of the problem, do not require explicit communication between the agents and can be implemented using a constant amount of memory resources. A schematic flow chart of the protocol is presented in Figure 19.

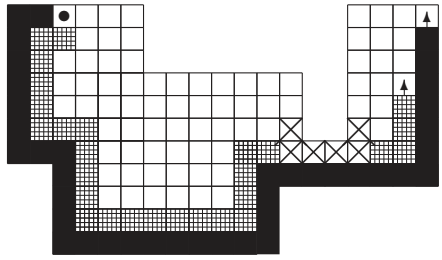

Fig. 10. An example of two agents using the **SWEEP** protocol, at time step 40 (with contamination spreading speed $d > 40$). All the tiles presented were contaminated at time 0. The black dot denotes the starting point of the agents. The X's mark the *critical points* which are not cleaned. The black tiles are the tiles cleaned by the first agent. The second layer of marked tiles represent the tiles cleaned by the second agent.

5.3. Cleaning Protocol: Definitions and Requirements

Definition 6. Let $\tau(t) = \left(\tau_1(t), \tau_2(t), \ldots, \tau_k(t)\right)$ denote the locations of the k agents at time t.

Definition 7. For a tile v, let $Neighborhood(v)$ denote the contamination states of v and its $8Neighbors$.

Let $\mathcal{M}_i$ denote some finite amount of memory contained in agent i, storing information needed for the protocol (e.g. the last moves of agent i). The requested cleaning protocol is therefore a rule f such that for every agent i

$$\tau_i(t+1) = f\left(\tau_i(t), Neighborhood(\tau_i(t)), \mathcal{M}_i(t)\right),$$

where the Manhattan distance between $\tau_i(t+1)$ and $\tau_i(t)$ is at most 1.

Definition 8. Let ∂F denote the boundary of F. A tile is on the boundary if and only if at least one of its $8Neighbors$ is not in F, meaning

$$\partial F = \{v \mid v \in F \ \wedge \ 8Neighbors(v) \cap (G \setminus F) \neq \emptyset\}.$$

The requested rule f should meet the following goals:

- **Successful Termination**: $\exists t_{\text{success}}$ s.t. $F_{t_{\text{success}}} = \emptyset$ (for as many initial instances of the problem as possible).
- **Agreement on Completion**: within a finite time after completing the mission, all the agents must halt.
- **Efficiency**: the cleaning process should be efficient in time and in agents' memory resources.
- **Fault Tolerance**: if one or several agents stop working ("die"), the rest of the agents will continue the cleaning process as efficiently as their number allows them.

5.4. Simple Connectivity Preserving Spreading

In Section 3 the spreading of the contamination was mentioned to optionally be defined in a variety of ways,

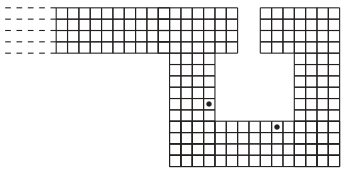

Fig. 11. A creation of a new hole around an agent. Notice how upon a contamination spread, the agents (denoted by black dots) may be trapped inside the "hole" that will be created. In this case, the agents will continue to clean the interior area of the hole endlessly. This problem is avoided by the elastic barrier.

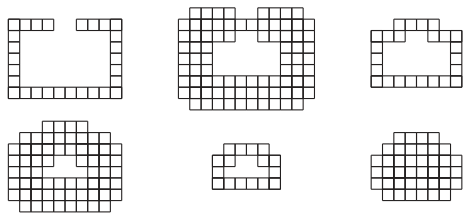

Fig. 12. An example of a *delaying ring*—a hole, which delays the cleaning operation of the agents. The left-hand region at the top represents the original contaminated region. The rest of the regions describe the spreading and cleaning processes. This phenomenon occurs while a region of clean tiles is surrounded by contaminated tiles due to a contamination spread. Left unattended, the clean region will eventually become contaminated, after additional contamination spreads take place (the exact number of the spreads needed depends on the width of this region). Notice though that should an agent reach this region after the first spread but before the clean region had become entirely contaminated, it will not be able to clean it, regardless of the number of "layers" pilled (as once a "ring" composed of the contaminated tiles surrounding the clean region is left, it is entirely composed of critical points, and therefore cannot be cleaned). The contaminated region surrounding the hole could therefore be cleaned only once the contamination process of the internal clean region is completed. This phenomenon is avoided by the elastic barrier.

controlled by the evolution of the edges E that form links between some (or all) of the pairs of adjacent contaminated tiles. In Definition 4 the edges of E were defined to connect each pair of contaminated tiles. This model, however, holds some perils which may interfere with the proper function of our local-information-dependent agents, implementing the **SWEEP** protocol—as, for such agents, preserving the simple connectivity of the contaminated region is crucial.

For example, the generation of holes in the region may prevent the agents from being able to accomplish the cleaning mission (see an example in Figure 11) or delay it significantly (see Figure 12 for a demonstration of the *delaying ring* phenomenon). Alternatively, cleaning critical points and transforming the region into several smaller sub-regions may generate a sub-region with no agents located in it—resulting in the termination of the cleaning mission by the agents, while some of the tiles are still contaminated.

We resolve this issue by altering the definition of the contamination spread, in a way which simulates an imaginary "elastic barrier" surrounding the contaminated region. This revised model, defined formally in Definition 9, is shown to prevent the creation of holes as a result of contamination spreading. This is complementary by the definitions of the cleaning protocol, which makes sure that no holes are generated by the agents' cleaning activities (as those are cleaning only boundary tiles, while also refraining from cleaning critical points).

The following is a detailed and formal definition of the revised model. The contaminated region F_t is assumed to be surrounded at its boundary by a "rubber-like" elastic barrier, dynamically reshaping itself to fit the evolution of the contaminated region over time (the barrier is derived from the edges of E_t, as demonstrated in Figure 13). This process can be thought of as a water-filed rubber balloon—while the shape of the water contained in it may change, as well as its volume, the balloon keeps bounding it tightly.

When an agent cleans a contaminated tile, the barrier is withdrawn, in order to free the tile. This is demonstrated in Figure 14.

While the contamination spreads (every d time steps), the elastic barrier stretches accordingly. First, all the clean tiles located to the right of a contaminated tile are becoming contaminated. Then, the clean tiles located

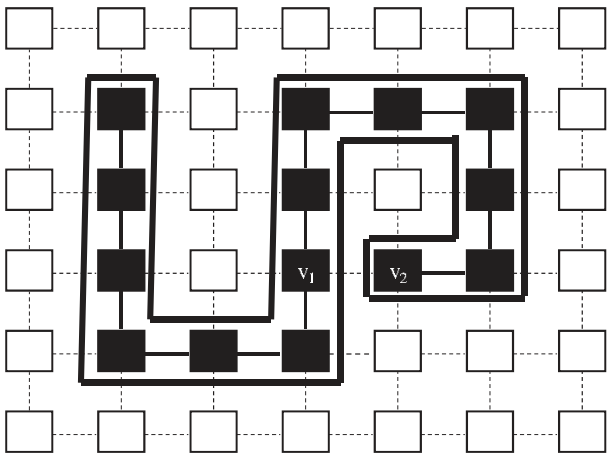

Fig. 13. A demonstration of the barrier surrounding the contaminated region, derived from the edges of E_t. Notice that when an edge connecting two contaminated tiles is not in E_t, those tiles are "on the other side" of the barrier. For example, observe the vertices v_1 and v_2, which are 4*Neighbors* in F_t (namely, the Manhattan distance between them in G equals 1); however, as $\neg((v_1, v_2) \in E_t)$, the geodesic distance between them is significantly greater.

below contaminated tiles, followed by clean tiles located to the left of such tiles. Finally, clean tiles located above a contaminated tile are affected. The barrier itself it implicitly derived from the edges connecting the tiles. Namely, it is the boundary which surrounds the connected contaminated

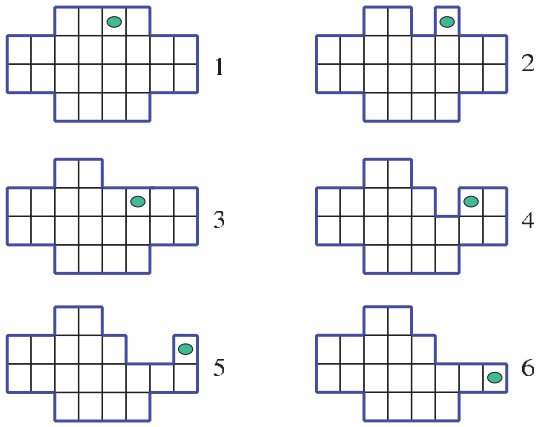

Fig. 14. A demonstration of the evolution of the elastic barrier as a result of the movement and cleaning of an agent according to the **SWEEP** protocol, as described in Figures 20 and 21.

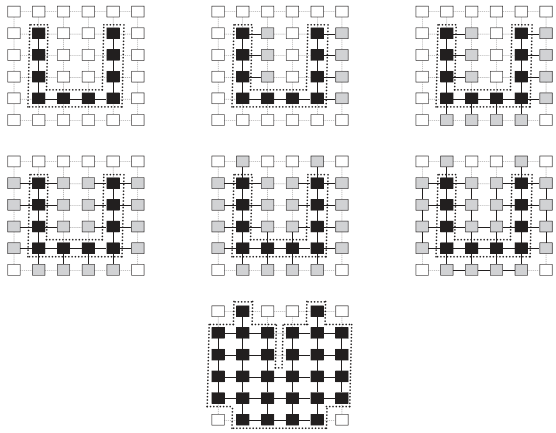

Fig. 15. A demonstration of the barrier expansion process as a result of a contamination spread.

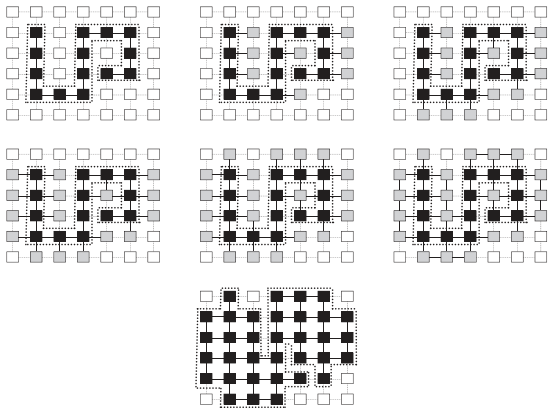

Fig. 16. A demonstration of the barrier expansion process as a result of a contamination spread.

tiles. This process is illustrated in Figures 15 and 16, and is formalized below.

Definition 9. Elastic barrier's expansion as a result of a contamination spread:

- Let $\widehat{V}_t = \emptyset$ be a set of vertices and let $\widehat{E}_t = \emptyset$ be a set of edges.
- For the vertex u located at (x_u, y_u), let:
  - $Neighbor_{\text{right}}(u)$ denote the vertex located at $(x_u + 1, y_u)$;
  - $Neighbor_{\text{left}}(u)$ denote the vertex located at $(x_u - 1, y_u)$;
  - $Neighbor_{\text{up}}(u)$ denote the vertex located at $(x_u, y_u + 1)$;
  - $Neighbor_{\text{down}}(u)$ denote the vertex located at $(x_u + 1, y_u - 1)$.
- Let $d_G(v, u)$ denote the number of edges between v and u in the graph G.
- Add new vertices to the right.
- $V_{\text{temp}} = \{v | \neg(v \in V_{F_t} \cup \widehat{V}_t) \wedge (\exists u \in V_{F_t}, v = Neighbor_{\text{right}}(u))\}$.
- $\widehat{E}_t \leftarrow \widehat{E}_t \cup \{(v, u) | (v \in V_{temp}) \wedge (v = Neighbor_{\text{right}}(u))\}$.
- $\widehat{E}_t \leftarrow \widehat{E}_t \cup \left\{ (v, u) \;\middle|\; \begin{array}{c} (v \in \widehat{V}_t) \wedge (v = Neighbor_{right}(u)) \wedge \\ (u \in V_{F_t}) \wedge (d_{G(V_{F_t} \cup \widehat{V}_t, E_t \cup \widehat{E}t)}(v, u) \leq 3) \end{array} \right\}$.
- $\widehat{V}_t \leftarrow \widehat{V}_t \cup V_{\text{temp}}$.
- Add new vertices to the bottom (repeat line 5.4 using $Neighbor_{\text{down}}$).
- Add new vertices to the left (repeat line 5.4 using $Neighbor_{\text{left}}$).
- Add new vertices to the top (repeat line 5.4 using $Neighbor_{\text{up}}$).
- $V_{F_t} \leftarrow V_{F_t} \cup \widehat{V}_t$.
- $E_t \leftarrow E_t \cup \widehat{E}_t$.
- $E_t \leftarrow E_t \cup \left\{ (v, u) \;\middle|\; \begin{array}{c} (v, u \in V_{F_t}) \wedge (v = 4Neighbors(u)) \wedge \\ (d_{F_t}(v, u) \leq 3) \end{array} \right\}$.

For the agents who travel along the tiles of F, the barrier "signals" the boundary of the contaminated region. When an agent detects a contaminated tile which is "on the other side" of the barrier, it treats it as though it was a clean tile (this aspect is important to remember when reviewing the agents' behavior while following the cleaning algorithm). For example, examining the region which appears in Figure 17 and assuming an agent located in the "X" marked tile, then, while looking "upwards", this agent acts as though the tile above it is clean (as the contaminated tiles are located behind the barrier and are thus masked as clean ones). Specifically, this is required for the proper execution of the "rightmost" function, discussed in Definition 11.

5.5. *The **SWEEP** Cleaning Protocol*

The **SWEEP** protocol is implemented by each agent a_i, located at time t at $\tau_i(t) = (x, y)$. We define below several terms we use while discussing the protocol. We

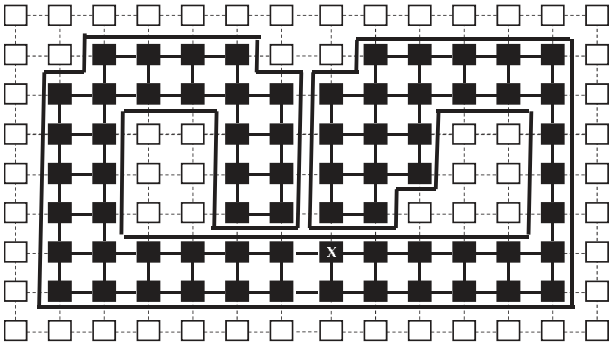

Fig. 17. Observe how the stretching of the barrier is generating "double fronts" after a contamination spread. Note that had the contaminated parts been allowed to merge, the simple connectivity of the region would have not been kept. In addition, observe how an agent located in the tile marked by an "X" is still kept on the boundary of the contaminated region, due to this mechanism.

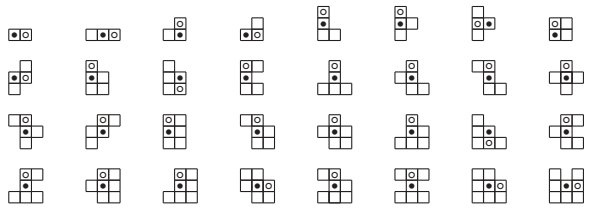

Fig. 18. When $t = 0$ the first movement of an agent located in (x, y) should be decided according to the initial contamination status of the neighbors of (x, y), as appears in these charts—the agent's initial location is marked with a filled circle while the destination is marked with an empty one. All configurations which do not appear in these charts can be obtained by using rotations. This definition is needed in order to initialize the traversal behavior of the agents in the correct direction.

stress the fact that this is indeed a *myopic* protocol, relying on neighborhood information, a *senile* protocol (the memory needed is constant) and it relies on *implicit local communication* only.

Definition 10. Let $\tilde{\tau}_i(t)$ denote the "previous location" of agent i. Namely, the last tile that agent i had been at, which is different from $\tau_i(t)$. This is formally defined as

$$\tilde{\tau}_i(t) \triangleq \tau_i(x) \text{ s.t. } x = \max\{j \in \mathbb{N} \mid j < t \text{ and } \tau_i(j) \neq \tau_i(t)\}.$$

Definition 11. The term "*rightmost*" is defined as follows:

- If $t = 0$, then select the tile as instructed in Figure 18.
- If $\tilde{\tau}_i(t) \in \partial F_t$, then, starting from $\tilde{\tau}_i(t)$ (namely, the previous boundary tile that the agent had been in), scan the *four neighbors* of $\tau_i(t)$ in a clockwise order until a boundary tile (excluding $\tilde{\tau}_i(t)$) is found.
- If $\tilde{\tau}_i(t) \notin \partial F_t$, then, starting from $\tilde{\tau}_i(t)$, scan the *four neighbors* of $\tau_i(t)$ in a clockwise order until the second boundary tile is found.

The additional information needed for the protocol and its sub-routines is contained in $\mathcal{M}_i$ and $Neighborhood(x, y)$.

A schematic flow chart of the protocol, describing its major components and procedures, is presented in Figure 19. The complete pseudo-code of the protocol and its sub-routines appears in Figures 20 and 21. Upon initialization of the system, the *System Initialization* procedure is called (defined in Figure 20). This procedure sets various initial values of the agents, and calls the protocol's main procedure—*SWEEP* (defined in Figure 21). This procedure, in turn, uses various sub-routines and functions, all defined in Figure 20. The *SWEEP* procedure comprises a loop which is executed continuously, until detecting one of two possible break conditions. The first, implemented in the *Check Completion of Mission* procedure, is in charge of detecting cases where all the contaminated tiles have been cleaned. The second condition, implemented in the *Check Near Completion of Mission* procedure, is in charge of detecting scenarios in which every contaminated tile contains at least a single agent. In this case, the next operation would be a simultaneous cleaning of the entire contaminated tiles. goes through the following sequence of commands. First, each agent calculates its desired destination at the current turn. Then, each agent calculates whether it should give a priority to another agent located at the same tile, and wishes to move to the same destination. When two or more agents are located at the same tile, and wish to move in the same direction, the agent who had entered the tile first gets to leave the tile, while the other agents wait. In the case where several agents had entered the tile at the same time, the priority is determined using the *Priority* function. Before actually moving, each agent who had obtained a permission to move must now locally synchronize its movement with its neighbors, in order to avoid simultaneous movements which may damage the connectivity of the region. This is done using the *waiting dependences* mechanism, which is implemented by each agent via an internal positioning of itself in a local ordering of its neighboring agents. When an agent is not delayed by any other agent, it executes its desired movement. It is important to notice that at any given time, *waiting* or *resting* agents may become active again, if the conditions which made them become inactive in the first place had changed.

5.6. Pivot Point

As in the **CLEAN** protocol, the initial location of the agents (the *pivot point*, denoted as p_0) is artificially set to be critical during the execution; hence, it is also guaranteed to be the last point cleaned. Completion of the mission can therefore be concluded from the fact that all (working) agents are back at p_0 with no contaminated neighbors to go to, thereby

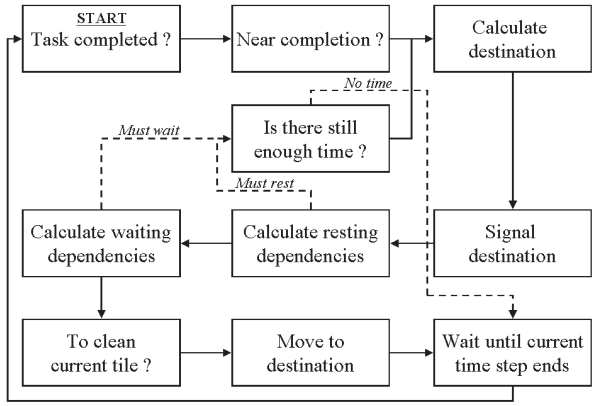

Fig. 19. A schematic flow chart of the **SWEEP** protocol. The smooth lines represent the basic flow of the protocol while the dashed lines represent cases in which the flow is interrupted. Such interruptions occur when an agent calculates that it must not move until other agents do so (as a result of either *waiting* or *resting* dependences—see lines 6 and 14 of **SWEEP** for more details).

```
1: System Initialization
2:     Arbitrarily choose a pivot point p_0 in ∂F_0, and mark it as a critical point
3:     Place all the agents on p_0
4:     For (i = 1; i ≤ k; i++) do
5:         Call Agent Reset for agent i
6:         Call SWEEP for agent i
7:         Wait two time steps
8:     End for
9: End procedure

10: Agent Reset
11:     resting ← false
12:     dest ← null /* destination */
13:     near completion ← false
14:     saturated perimeter ← false
15:     waiting ← ∅
16: End procedure

17: Priority
18:     /* Assuming the agent moved from (x_0, y_0) to (x_1, y_1) */
19:     priority ← 2(x_1 − x_0) + (y_1 − y_0)
20: End procedure

21: Check Completion of Mission
22:     If ((x, y) = p_0) and (x, y) has no contaminated neighbors, then
23:         If (x, y) is contaminated, then
24:             Clean (x, y)
25:         STOP
26: End procedure

27: Check "Near Completion" of Mission
28:     /* Cases where every tile in F_t contains at least a single agent */
29:     near completion ← false
30:     If each of the contaminated neighbors of (x, y) contains at least one agent,
        then
31:         near completion ← true
32:     If each of the contaminated neighbors of (x, y) satisfies near completion,
        then
33:         Clean (x, y) and STOP
34:     /* Cases where every non-critical tile in ∂F_t contains at least two agents */
35:     saturated perimeter ← false
36:     If ((x, y) ∈ ∂F_t) and both (x, y) and all of its non-critical neighbors
        in ∂F_t contain at least two agents, then
37:         saturated perimeter ← true
38:     If ((x, y) ∈ ∂F_t) and both (x, y) and all of its neighbors in ∂F_t have
        saturated perimeter = true, then
39:         Ignore resting commands for this time step
40: End procedure
```

Fig. 20. The first part of the **SWEEP** cleaning protocol. The term ∂F is defined in Section 5.3.

reporting on completion of their individual missions. Note that this artificial preservation of the criticalness of the pivot point is not necessary for the algorithm to work. However, it makes life easier for the user, as one can now know where to find the agents after the cleaning has been completed. If we do not start with all agents at the pivot and force p_0 to be critical, the location of the agents upon completion will generally not be predictable in advance.

5.7. Signaling

Since by assumption an agent's sensors can detect the status of all tiles which are contained within a digital sphere of radius 4 placed around the current location of the agent, each agent can artificially calculate the desired destination of all the agents which are located in one of its 4*Neighbors* tiles (see Figure 22). Thus, the *signaling* action of each agent can be simulated by the other agents near him, and hence explicit signaling by the agents is not actually required. However, the signaling action is kept in the description and flow chart of the protocol (in line 5 of the main procedure of the protocol) for the sake of simplicity and understandability.

5.8. Connectivity Preservation

The connectivity of the region yet to be cleaned, F, is preserved by allowing the agents to clean only *non-critical* points. This guarantees both successful termination and agreement of completion (since having no contaminated neighbors implies that $F = \emptyset$). Also, should several agents malfunction and halt, as long as there are still functioning agents, the mission will still be carried out, albeit at a slower pace (as long as the number of active agents is high enough).

Apart from the cleaning performed by the agents, tiles may become contaminated due to contamination spread. However, when two parts of the contamination region spread towards one another, the region's simple connectivity is still preserved, due to the basic assumption of the elastic barrier bounding the shape.

Note that when several agents are located in the same tile, only the last one who exits cleans it (see line 28 of the main procedure of the **SWEEP** protocol), in order to prevent agents from being "thrown" out of the contaminated region. An alternative method for ensuring that the agents are always capable of executing the cleaning protocol would have been the implementation of a "contamination-seeking mechanism" (i.e. by applying methods such as suggested in Baeza-Yates and Schott (1995)). Such a mechanism would have allowed an agent to clean its current location, even if other agents had also been present there. That solution,

1: **SWEEP Protocol** /* Controls agent i after **Agent Reset** */
2: **Check Completion of Mission**
3: **Check "Near Completion" of Mission**
4: $dest \leftarrow$ *rightmost neighbor* of (x,y) /* Calculate destination */
5: *destination signal bits* $\leftarrow dest$ /* Signaling the desired destination */
6: /* Calculate resting dependences (solves agents' clustering problem) */
7: From all agents in (x,y) except agent i we define the following groups:
8: K_1: agents signaling towards $dest$ which entered (x,y) before agent i
9: K_2: agents signaling towards $dest$ which entered (x,y) with agent i and with higher *priority* than that of agent i
10: $resting \leftarrow false$
11: **If** $(K_1 \neq \emptyset)$ or $(K_2 \neq \emptyset)$, then
12: $resting \leftarrow true$
13: **If** current time step $\mathcal{T}$ did not end yet, then jump to 4 **Else** jump to 30
14: $waiting \leftarrow \emptyset$ /* Waiting dependences (agents' synchronization) */
15: Let *active agent* denote a *non-resting* agent which did not move in $\mathcal{T}$ yet
16: **If** $(x-1,y) \in F_t$ contains an active agent, then $waiting \leftarrow waiting \cup \{left\}$
17: **If** $(x,y-1) \in F_t$ contains an active agent, then $waiting \leftarrow waiting \cup \{down\}$
18: **If** $(x-1,y-1) \in F_t$ contains an active agent, then $waiting \leftarrow waiting \cup \{l\text{-}d\}$
19: **If** $(x+1,y-1) \in F_t$ contains an active agent, then $waiting \leftarrow waiting \cup \{r\text{-}d\}$
20: **If** $dest = right$ and $(x+1,y)$ contains an active agent j, and $dest_j \neq left$, and there are no other agents delayed by agent i (i.e. $(x-1,y)$ does not contain active agent l with $dest_l = right$ and no active agents in $(x,y+1),(x+1,y+1)$, $(x-1,y+1)$, and $(x+1,y)$ does not contain active agent n with $dest_n = left$), then $(waiting \leftarrow waiting \cup \{right\})$ and $\left(waiting_j \leftarrow waiting_j \setminus \{left\}\right)$
21: **If** $dest = up$ and $(x,y+1)$ contains an active agent j, and $dest_j \neq down$, and there are no other agents delayed by agent i (i.e. $(x,y-1)$ does not contain active agent l with $dest_l = up$ and no active agents in $(x+1,y),(x+1,y+1)$, $(x-1,y+1)$, and $(x,y+1)$ does not contain active agent n with $dest_n = down$), then $(waiting \leftarrow waiting \cup \{up\})$ and $\left(waiting_j \leftarrow waiting_j \setminus \{down\}\right)$
22: **If** $(waiting \neq \emptyset)$, then
23: **If** $\mathcal{T}$ has not ended yet, then jump to 4 **Else** jump to 30
24: /* Decide whether or not (x,y) should be cleaned */
25: **If** $\neg\left((x,y) \in \partial F_t\right)$ or $\left((x,y) \equiv p_0\right)$ or (x,y) has two contaminated tiles in its 4*Neighbors* which are not connected via a path of contaminated tiles from its 8*Neighbors*, then
26: (x,y) is an *internal point* or a *critical point* and should not be cleaned
27: **Else**
28: Clean (x,y) if and only if it still does not contain other agents
29: Move to $dest$
30: Wait until $\mathcal{T}$ ends
31: Return to 2.

Fig. 21. The **SWEEP** cleaning protocol. The term *rightmost neighbor* is defined in Section 5.5. *l-d* and *r-d* are left-down and right-down, respectively.

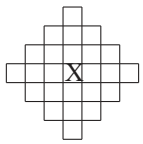

Fig. 22. Digital sphere of diameter 7, placed around the agent.

however, would have been even less elegant and would have added additional difficulties to the analysis process.

5.9. Agents' Synchronization

Note that agents operating in the described environment must have some means of synchronization, which is necessary in order to prevent agents from operating at the same time—risking cutting the contamination region into several connected components, as shown in Figure 23.

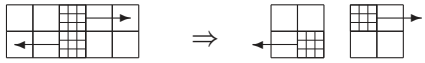

Fig. 23. When the agents do not possess a synchronization mechanism, they may, among others, damage the region's connectivity. In this example, two agents clean their current locations, and move according to the **SWEEP** protocol. Since they are not synchronized, the tiles which they are located in are not treated as critical at the times of cleaning. However, the region's connectivity is not preserved. Should one of the agents had waited for its neighbor to complete executing the protocol's steps before resuming its actions, while deciding whether to clean its current location, it would have treated this tile as critical, and therefore avoid cleaning it. In this case, the connectivity of the region would have been maintained.

To ensure that such scenarios will not occur, a local order between the operating agents must be implemented. Note that—throughout the next paragraphs—agents which are signaling a *resting* status (see Section 5.10 for more details) are being disregarded while calculating the dependences of the agents' movements. Note also that the calculation of *resting* status of each agent can be calculated several times during each time step, as movements of agents may cause this property to change. The creation of the following order is implemented in lines 14 through 21 of **SWEEP**:

Definition 12. For agent i, let

$$P_i \subseteq \{up,\ down,\ left,\ right,\ right\text{-}down,\ left\text{-}down\}$$

be a set of directions of tiles, in which there are currently agents, which agent i is delayed by (meaning that agent i will not start moving until the agents in these tiles move). Unless stated otherwise, $P_i = \emptyset$.

For agent i which is located in tile (x,y), if $(x-1,y)$ is a tile in F, which contains an agent, then $P_i \leftarrow P_i \cup \{left\}$. If $(x,y-1)$ is a tile in F, which contains an agent, then $P_i \leftarrow P_i \cup \{down\}$, and similarly for $(x+1,y-1)$ and *right-down* and for $(x-1,y-1)$ and *left-down*.

Definition 13. Let $dest_i \in \{up,\ down,\ left,\ right\}$ be the destination agent i might be interested in moving to, after leaving its current location.

We assume that agents can sense the desired destinations of adjacent agents. For example, each tile can be treated as a physical tile, in which the agents can move. Thus, by approaching the side of the tile it is planning to move to, an agent can "communicate" this two-bit information to surrounding agents. Alternatively, this can be implemented using any other hardware or software capabilities that may be available to the agents.

Let each time step be divided into two phases. In phase 1, every agent "signals" the destination it intends to move towards, either by moving to the appropriate side of the tile, or by using the destination flag.

Since we defined an artificial rule which states the superiority of *left* and *down* over *right* and *up* (and, internally, of *down* over *left*), there are several specific scenarios in which this asymmetry should be reversed in order to ensure a proper operation of the agents. This "dependences switching" rule is defined as follows.

For an agent i located in (x,y) and intending to move to the right (namely, $dest_i = right$), and where *all* the following hold:

- Tile $(x+1,y)$ contains an agent j, where $dest_j \neq left$.
- There are no other agents which are delayed by agent i. Namely:
  - Tile $(x-1,y)$ does not contain an agent l, where $dest_l = right$.
  - Tiles $(x,y+1),(x+1,y+1),(x-1,y+1)$ do not contain any agent.
  - Tile $(x+1,y)$ does not contain an agent n, where $dest_n = left$),

then

$$P_i \leftarrow P_i \cup \{right\}, \qquad P_j \leftarrow P_j \setminus \{left\}.$$

Similarly, for an agent i located in (x,y) and going up ($dest_i = up$), and where *all* the following hold:

- Tile $(x,y+1)$ contains an agent k, and $dest_k \neq down$.
- There are no other agents which are delayed by agent i. Namely:
  - Tile $(x,y-1)$ does not contain an agent m, where $dest_m = up$.
  - Tiles $(x+1,y),(x+1,y+1),(x-1,y+1)$ do not contain any agent.
  - Tile $(x,y+1)$ does not contain an agent q, where $dest_q = down$,

then

$$P_i \leftarrow P_i \cup \{up\}, \qquad P_k \leftarrow P_k \setminus \{down\}.$$

At phase 2 of each time step, the agents start to operate in turns, according to the order implied by P_i. This guarantees that the connectivity of the region is kept, since the simultaneous movement of two neighboring agents is prevented.

Notice that deadlocks are impossible—since the basic rule is that every agent is delayed by the agents in its *left* and *down* neighbor tiles. Therefore, at any given time, and for every possible group of agents, there exists an agent with the minimal x and y coordinates (which, by definition, is not delayed by any other agent of this group). After this agent moves, all the agents which are delayed by it can now move, and so on. As to the "dependences switching rule"—let agent i located in tile (x,y) have the minimal x and y values among the agents who had not moved yet, let $dest_i = up$ and let tile $(x,y+1)$ contain an agent j such that $dest_j \neq down$. Then, although agent i is located below agent j, it will be delayed by it (i.e. $(up \in P_i)$ and $\neq(down \in P_j)$) as long as agent i is not delaying any other agent (as this is the requirement of the "dependences switching rule"). In this case, we should show that there cannot be a cycle of dependences, which starts at agent j, ends at agent i and is closed by the dependency of agent i on agent j. Such a cycle cannot exist since for it to end in agent i, it means that agent i is delaying another agent k. However, this is impossible since agent i is known not to delay any agent (specifically, agent k). Hence, circular dependences are prevented and no deadlocks are possible.

Note again that while phase 2 is in process, F_t may change due to cleaning by the agents. As a result, the desired destinations of the agents as well as their dependency graphs must also be dynamic. This is achieved through the repeated recalculation of these values, by every *waiting* agent. For example, assume agent i to be located in (x,y), and $dest_i = down$ without loss of generality, and let agent j located in tile $(x,y-1)$ move out of this tile and clean it. Then, $dest_i$ naturally changes (as the tile (x,y) does no longer belong to F_t, and thus is not a legitimate destination for agent i), and, thus, P_i may also change. In this case, agent i should change its "destination signal" and act according to its new $dest_i$ and P_i. This is implemented in **SWEEP** by calculating the waiting agents' destinations and dependences lists repeatedly, until either all agents have moved or until the time step has ended (meaning that some agents had to change their status to *resting*, and pause until the next time step—see Section 5.10 for more details). Note that every *waiting* agent is guaranteed either to complete its movement in the current time step, or to be forced to wait for the next time step, by switching its status to *resting*.

Note that the goal of the inherent dominance of *up* over *down* and of *left* over *right* is to act as a kind of "semaphore", preventing scenarios in which adjacent tiles are cleaned at the same time, destroying the connectivity of the region (as demonstrated in Figure 23). The goal of the "dependences switching rule" on the other hand is simply to avoid the phenomenon demonstrated in Figure 24, which may decrease the cleaning efficiency of the agents. Therefore, note that the "dependences switching rule" is in fact not required in order to ensure a proper completion of the mission, but rather to improve the agents' performance.

5.10. Clustering Problem

Since we are interested in preventing the agents from moving together and functioning as a single agent (which leads to an obvious decrease in the system's performance), we would like to impose a "resting policy" which will ensure that the agents do not group into clusters which move together. This is done by artificially delaying agents in their current locations, should several agents placed in the same tile wish to move to the same destination. However, we would like the resting time of the agents to be minimal, for performance and analysis reasons.

The following resting policy is implemented by lines 6 through 13 of the **SWEEP** protocol. Using its sensors, an

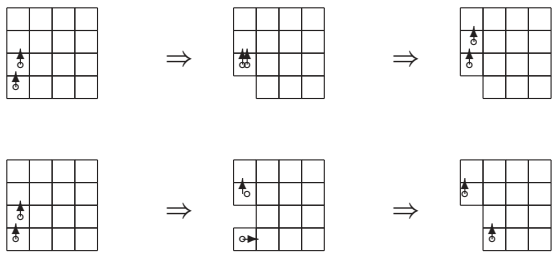

Fig. 24. The upper charts demonstrate a performance bug which may be caused due to local dependences. The agents are advancing according to the **SWEEP** protocol, but their cleaning performance is decreased. The lower charts demonstrate the cleaning operation after adding the dependences switching mechanism.

agent intending to move into tile v is aware of other agents which intend to move into v as well. Thus, when an agent enters a tile which already contains other agents, it can determine whether these agents entered this tile at the same time step that it did, or whether they had occupied this tile before the current time step had started (note that this can be implemented without assigning any IDs to the agents, but rather by observing only the movements of other agents in the agent's vicinity).

Note that in phase 1 of each time step all the agents are signaling their desired destinations. Thus, an agent can identify which ones of the agents which are located in its current tile intend to move to the same destination as its own. Only those agents may cause this agent to delay its actions and rest.

From the agents which intend to move in the same direction as agent i, the agent can distinguish between three groups of agents—the agents which entered this tile before the time step agent i did (group A_2), those which entered the tile in the same time step as i (group A_4) and those who entered it after agent i did (group A_3). Again, this is implemented without assigning unique IDs to the agents but, rather, by observing their movements and merely *counting* the number of agents at each group. If $A_2 = \emptyset$, agent i waits, changes its status to *resting agent*, signals this status to the other agents in its vicinity and does not move. As this rule is kept by every other agent, agent i is guaranteed that the agents of group A_3 in their turn will wait for agent i to move before being free to do so as well.

As to the agents of A_4, note that at any given time, two agents cannot leave the same tile in the same direction at the same time step. For small values of t (i.e. at the beginning of **SWEEP**) as the agents are periodically released by the initialization procedure of **SWEEP**, no two agents are released at the same time step. Later, since we are assured that all the agents of group A_4 arrived to v from different tiles (which are also different from the tile agent i had entered v from), and since all the agents in group A_4 know the previous locations of each other, a consensus over a local ordering of A_4 can be established (note that no explicit communication is needed to form this ordering). An example for such an order is the **Priority** function of the **SWEEP** protocol. As a result, the agents are able to exit the tile they are currently located in, in an orderly fashion, according to a well-defined order. Hence, the following invariant holds: "*at any given time t, for any two tiles v, u, there can only be a single agent which moves from v to u at time step t*". Therefore, the clustering problem is solved.

Notice that there is a single exception to this mechanism, in which the resting commands are overruled. This happens when all non-critical perimeter tiles contain at least two agents. In this scenario, following the resting commands would have created a situation in which no tile can be cleaned, as for every agent which leaves a certain tile there are still "resting" agents located in the same tile. Therefore, when this scenario is detected by the *Check Near Completion of Mission* procedure of **SWEEP**, the agents ignore their resting commands momentarily, in order to be able to clean the non-critical perimeter tiles. The order and internal prioritization of the agents are maintained, for calculating the new *resting* commands in the following time step.

5.11. Mission Termination

The termination of the protocol is done in one of two cases—either an agent finds itself in the pivot point, while all of its neighbors are clean (which means that this is the last contaminated tile and therefore should be cleaned), or when each contaminated tile contains at least one agent (which is a generalization of the previous scenario). The second case is implemented by allowing the agents to signal to their four neighbors whether all of their contaminated neighbors contain at least one agent.

6. Upper Bounds over the Cleaning Time of the SWEEP Protocol

6.1. Overview of the Results

As already stated previously, since we have no easy way to decide whether k agents can successfully clean an instance of the *dynamic cooperative cleaners* problem, producing bounds for the proposed cleaning protocol is important for estimating its efficiency. In this section, we discuss the efficiency of the **SWEEP** cleaning protocol, by analyzing its time complexity and producing upper bounds on the cleaning time of a group of k agents. Analyzing the performance of the above-defined protocol is quite difficult. Due to the preservation of the *critical points*, such points can be visited many times by the agents before being cleaned. Furthermore, due to the dynamic nature of the problem, the shape of the contaminated region can change dramatically during the cleaning process. This section is organized as follows: a recursive upper bound over the cleaning time of a group of agents appears in Theorem 4, while various direct approximations for this bound are discussed in Theorems 5 and 6. An upper bound for the minimal number of agents required in order to guarantee a

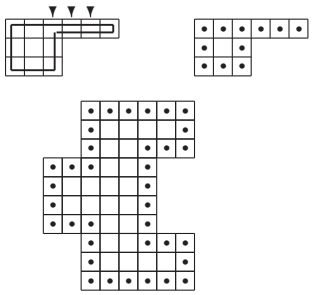

Fig. 25. The line in the left upper chart goes through the tiles of C_t where the three arrows denote tiles that are included twice. The circles in the right upper chart denote the tiles of ∂F_t. Note that while ∂F_t contains 11 tiles, C_t contains 14. In the lower chart, there are no *critical points* in ∂F_t and therefore $c_t = |\partial F_t| = |C_t| = 36$.

successful cleaning of a given contaminated region is then derived and presented in Theorem 7.

6.2. Definitions

Recalling Definitions 5 and 8, S_t denotes the size of the contaminated region F at time t while the boundary of the contaminated region F is denoted as ∂F. Let d denote the number of time steps between two contamination spreads.

Definition 14. A *path* in F is defined to be a sequence $(v_0, v_1, \ldots, v_n)$ of tiles in F such that every two consecutive tiles are *4 connected* (the *Manhattan* distance between them is 1). The *length* of a *path* is defined to be the number of tiles in it.

Definition 15. Let tile v be called a *critical point* if there exist $v_1, v_2 \in 4Neighbors(v)$ for which all paths connecting v_1 and v_2, included in $8Neighbors(v)$, necessarily pass through v (where v, v_1, v_2 and all said paths are in F).

Definition 16. We shall denote by c_t the circumference of F at time t, defined as follows: let v_0 and v_n be two *4 connected* tiles in ∂F_t, and let $C_t = (v_0, v_1, \ldots, v_n)$ be a shortest *path* connecting v_0 and v_n, which contains all the tiles of ∂F_t and only such tiles.

Notice that C_t may contain several instances of the same tile, if this tile is a *critical point*(meaning that C_t is an ordering of the tiles of ∂F_t, in which multiple instances of tiles that are *critical points* are allowed). c_t will be defined as the length of C_t. An example appears in Figure 25.

Note that the *cardinality* of C_t equals $|\partial F_t|$, in which every tile can count at most once.

Definition 17. For some $v \in F_t$, let $Strings_t(v)$ denote the set of all *paths* in F_t that begin in v and end at any *non-critical point* in ∂F_t, and let $w(F_t, v)$ denote the *depth* of v—the length of the shortest *path* in $Strings_t(v)$ (unless v is a *critical point*, in which case its *depth* is defined to be zero).

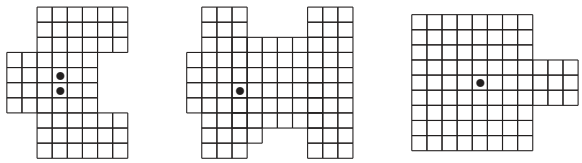

Fig. 26. The chart contains three regions. The black circles denote the deepest tiles within the regions. Notice that while in the left-hand region there are two tiles whose depth is maximal, in the other regions there is only one tile of maximal depth. The widths of the regions equal the depth of the deepest tiles, and are (from left to right) 2, 3 and 4.

Definition 18. Let $W(F_t)$ denote the *width* of F_t, defined as the maximal depth of all the tiles in F_t, i.e.

$$W(F_t) = \max\{w(F_t, v) \mid v \in F_t\}.$$

An example appears in Figure 26.

Definition 19. Let F'_t denote the region one would obtain, having one agent traverse F_t once, using the **SWEEP** protocol (i.e. when $k = 1$, $F'_t = F_{t+c_t}$). For the sake of simplicity, F''_t will be used instead of $(F'_t)'$, and so on.

Definition 20. The longest in-region distance between p_0 and any other point in F_t will be referred to as $l(F_t)$, the *length* of F_t:

$$l(F_t) = \max_{v \in F_t} \left\{ d_{F_t}(p_0, v) \right\},$$

where $d_{F_t}(x, y)$ is the distance (shortest path within F_t) between x and y.

6.3. Recursive Upper Bound

The complete proofs of the following lemmas can be found in Wagner et al. (2008). The lemmas are reiterated here for the sake of completeness.

Lemma 1. *The cardinality of the region's circumference always decreases after being traversed by an agent applying the **SWEEP** protocol (assuming no spread occurs), namely*

$$|\partial F'| \leq \max\{1, |\partial F| - 8\}.$$

Proof. For a proof that applies for any region F such that the number of tiles in F' is at least two, see Lemma 3 in Wagner et al. (2008). This proof, though, does not always hold for regions that, after being traversed once, produce either $\emptyset$ or a single tile. The reason is that the proof of the lemma that appears in Wagner et al. (2008) relies on the fact that for each "turn" in F, there is a corresponding "turn" in F'. However, the concept of "turn" requires the

existence of a possible movement of an agent from one tile to another tile. This obviously cannot be done for $F' = \emptyset$ or for $F' = \{v\}$. For such regions it can easily be seen that the lemma holds, as $|\partial\emptyset| = 0$ and as $|\partial\{v\}| = 1$. □

Lemma 2. *Every time a region is traversed by an agent using the **SWEEP** protocol, its width decreases by at least 1, if no spread occurs, namely*

$$W(F'_t) \leq W(F_t) - 1.$$

Proof. See Lemma 4 in Wagner et al. (2008). □

Lemma 3. *The time it takes an agent which uses the **SWEEP** protocol to move along a path of length c_t (including delays caused by other agents located in the same tiles) is at most $4 \cdot c_t$.*

Proof. See Lemma 5 in Wagner et al. (2008). □

Lemma 4. *c_t, the length of the circumference of F_t, never exceeds twice its cardinality, namely*

$$c_t \leq 2 \cdot |\partial F_t| - 2.$$

Proof. See Lemma 6 in Wagner et al. (2008). □

An important feature of a region F_t is its size—a bound of the contaminated region's size will later be used for producing an upper bound for the cleaning time of the agents. The following lemma presents an upper bound for the size of the contaminated region at time t.

Lemma 5. *At any given time, the size of the contaminated region (i.e. $S_t = |F_t|$) can be bounded as follows:*

$$S_t \leq S_0 + \eta \cdot c_0 + 2(\eta^2 + \eta), \qquad \text{where} \qquad \eta \triangleq \left\lfloor \frac{t}{d} \right\rfloor.$$

Proof. The only time clean tiles can become contaminated is when the contamination spreads. When producing an upper bound for S_t, we can therefore disregard the cleaning done by the agents, as it can never result in the contamination of clean tiles. The maximal number of new contaminated tiles is achieved when the already contaminated tiles have the maximal boundary area possible. It is easy to see that this happens when the tiles are arranged in the form of a straight line. Since F_t is connected, it has at least $S_t - 1$ edges, meaning that the sum of the degrees of the tiles is at least $2(S_t - 1)$. Thus, the maximal boundary area is at most $4S_t - 2(S_t - 1)$ (four possible neighbors per contaminated tile minus the edges already connecting the contaminated tiles). In a straight line, all tiles but two have two clean neighbors while two tiles have three clean neighbors. This sums up to $2(S_t - 2) + 6$, which is exactly the maximal boundary area possible.

In such a case, when the contamination spreads for the ith time, the area of the region is increased by $c_0 + 4i$. Let S_{s_i} denote the area of F_t after the ith spread. Then, for any region F_t, this is bounded as follows:

$$S_{s_i} \leq S_0 + i \cdot c_0 + 4(1 + 3 + 5 + \ldots + i).$$

The last inequality can be simplified to

$$S_{s_i} \leq S_0 + i \cdot c_0 + 2(i^2 + i). \tag{3}$$

Since a spread occurs every d time steps, t can be written in the form of: $t = \eta \cdot d + \alpha$, where η is the number of spreads and α the remainder. Since we are only interested in the number of spreads, we can use

$$\eta = \left\lfloor \frac{t}{d} \right\rfloor.$$

By applying this to Equation (3), we get the requested result. □

It is obvious that the number of tiles in the boundary of F_t is equal to or smaller than its total size, namely

$$\partial F_t \leq S_t.$$

After combining this with Lemmas 4 and 5 and with the "intra-spreads $|\partial F_t|$ preservation" property of Lemma 1, we produce the following bound for c_t.

Lemma 6. *At any given time, the circumference of the contaminated region F_t is bounded as follows:*

$$c_{t+1} \leq 2\Big[\Lambda_t + \eta \cdot c_0 + 2(\eta^2 + \eta) - 1\Big],$$

where

$$\eta \triangleq \left\lfloor \frac{t}{d} \right\rfloor \qquad \text{and} \qquad \Lambda_t = \begin{cases} |\partial F_0|, & t \leq d, \\ S_0, & t > d. \end{cases}$$

Proof. According to Lemma 4, between two spreads, c_t is bounded by twice its original value (from just after the last spread). After a spread occurs, the value of c_t is being reset, according to the bound for the total size of F_t (Lemma 5). □

It is also easy to see that the width of F at any time is bounded by its initial width plus the number of contamination spreads. This notion is expressed in the following lemma.

Lemma 7.

$$W(F_{t+1}) \leq W(F_t) + 1 \qquad \text{for} \qquad t \mod d = 0,$$
$$W(F_{t+1}) \leq W(F_0) + \eta, \qquad \text{where} \qquad \eta \triangleq \left\lfloor \frac{t}{d} \right\rfloor.$$

With $W_{\text{REMOVED}}(t)$ denoting the decrease in F's width due to the agents' cleaning activity until time t, we can now show the following lemma.

Lemma 8.

$$\text{If} \quad \left\lfloor \sum_{i=1}^{t} \frac{k}{4 \cdot c_i} \right\rfloor \geq \overline{W}(F_{t+1}),$$

$$\text{then} \quad W_{\text{REMOVED}}(t) \geq \overline{W}(F_{t+1}),$$

where $\overline{W}(F_{t+1})$ *denotes the width of* F_0 *at time* $t+1$*, assuming that no cleaning took place by the agents thus far (namely, the width predicted by Lemma7).*

Proof. $W_{\text{REMOVED}}(t)$ can be defined as the number of completed traversals around the contaminated region, multiplied by the number of the agents k. Note that at time step t, each agent by definition completes $\frac{1}{c_t}$ of the circumference. Since we know the value of c_t for every t, and since we know that the time it takes an agent to move along a path is at most four times the length of this path (see Lemma 3), then the decrease in the original width of the region caused by the cleaning process of the agents is bounded as described above.

Note that as the expression which appears in the lemma does not distinguish between a multitude of agents working for a short while, and a single agent working for a long period of time, it may not hold for smaller values of t (in which the accumulated partial peelings is smaller than $\overline{W}(F_{t+1})$). Therefore, while writing $W_{\text{REMOVED}}(t) \geq \left\lfloor \sum_{i=1}^{t} \frac{k}{4 \cdot c_i} \right\rfloor$ might not always hold, the expression as appears above does.

Note that this holds also when taking into account the effect the spreading contamination has on the region's width. Let $p = (v_1, v_2, \ldots, v_n)$ denote a "tour", traversing a *contaminated* region F, and let us assume that a spread occurs at time $t = t_s$. Let $p = p_1 \cdot p_2$, and let p_1 denote the part of the "tour" which took place before the contamination spread.

The width of the region F is defined as the depth of the deepest tile in F. Let us assume without loss of generality that there is only one tile of maximal depth, u (if there are more than a single deepest tile, the same observation can be applied to all deepest tiles, separately). Therefore, $W(F_t)$ is the length of the shortest *path* (or paths) from u to a *non-critical* tile in ∂F_t. Let us assume without loss of generality that there is only one such shortest path, from u to a *non-critical* boundary tile u_{END} (if there are more than a single shortest path, the same observation can be applied to all, separately). u_{END} is a boundary tile, and therefore must be included at least once in p (either in p_1 or in p_2, or in both). If $(u_{\text{END}} \in p_1) \wedge (u_{\text{END}} \in p_2)$, then u_{END} would be a *critical point*, and we already know this is not the case.

If $(u_{\text{END}} \in p_1)$, then it was already cleaned prior to t_s, decreasing the width of the region by 1. After the contamination spread, u_{END} might become *contaminated* once again, restoring the region's width to its original value.

If $\neg(u_{\text{END}} \in p_1)$, it therefore means that $(u_{\text{END}} \in p_2)$. As u_{END} is a boundary tile of F, it has at least one 8*Neighbour* which is clean prior to t_s. In order for the contamination spread to increase the width of F, it means that all the 8*Neighbors* of u_{END} which were clean have become contaminated after the spread. However, as those neighbors are perimeter tiles, they will be visited by the relevant agent. As at least one of them can be cleaned (since there cannot exist a tile whose 8*Neighbors* are all critical points), u_{END} will become a perimeter tile again, restoring the original value of the width of F.

Spreads can naturally occur more than once in the middle of a "tour". In this case, the same principle can be recursively applied. An example of the above-described situation appears in Figure 27. □

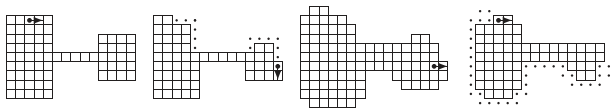

Fig. 27. An example of an "interrupting spread"—a spread that occurs in a middle of a "tour". The first chart on the left is the original contaminated region, whose width is 3 (the black circle is the relevant agent and the arrow marks its next movement). The second chart is the region just before the spread (the dots are the cleaned tiles). The third chart is the region just after the spread (notice that the width is now 4 and that the "tour" was changed). The fourth chart is the region after the agent completes its (new) "tour". Notice that the width is 3 again.

A rough upper bound over the cleaning time of a contaminated region F_0 can now be derived, as follows.

Theorem 4. *Assume that* k *agents start cleaning a simply connected region* F_0 *at some boundary point* p_0 *and work according to the* ***SWEEP*** *protocol, and denote by* $t_{\text{success}}(k)$ *the time needed for this group to clean* F_0*. Then we have:*

1. *If* $(t = \frac{8(|\partial F_0|-1)\cdot(W(F_0)+k)}{k} + 2k)$ *is smaller than or equal to* d*, then* $t_{\text{SUCCESS}} = \lceil t \rceil$*. This also holds for static environments, since in such cases* $d \to \infty$*.*
2. *Otherwise* $(t > d)$*:* t_{SUCCESS} *is the minimal (integer)* t *for which*

$$\sum_{i=d+1}^{t} \frac{1}{S_0 - 1 + 2\lfloor \frac{i}{d} \rfloor^2 + (c_0 + 2)\lfloor \frac{i}{d} \rfloor} \geq \gamma + \frac{8}{k} \cdot \left\lfloor \frac{t}{d} \right\rfloor,$$

where

$$\gamma \triangleq \frac{8(k + W(F_0))}{k} - \frac{d - 2k}{|\partial F_0| - 1}.$$

Proof. In order for F_0 to be cleaned, there should exist a time t_{SUCCESS} in which the width of the region will be 0. Remembering that $W(F_t)$ is monotonically increasing and disregards the cleaning performed by the agents, and that $W_{\text{REMOVED}}(t)$ denotes the decrease in the width of F_0 due to the cleaning of the agents, we would like the upper bound over the width of the region to be smaller than the width's

decrease. Namely, we are interested in

$$W(F_{t_{\text{SUCCESS}}}) + k \le W_{\text{REMOVED}}(t_{\text{SUCCESS}}).$$

The purpose of the "$+k$" is to guarantee the cleaning of the "skeleton" that remains when the width of the region decreases to zero. In such a time, the agents function as a single agent (and therefore we add the "k", which symbolizes a group of k agents acting as one), and we must guarantee that an additional traversal will be performed. We now apply Lemmas 7 and 8.

Note that either $t_{\text{SUCCESS}} \le d$ or $t_{\text{SUCCESS}} > d$. We first examine the case of $t_{\text{SUCCESS}} \le d$. Regarding $W(F_t)$, c_t and $W_{\text{REMOVED}}(t)$, this means that

$$\forall i \le t, \quad W(F_i) \le W(F_0),$$

$$\forall i \le t, \quad c_{i+1} \le 2(|\partial F_0| - 1),$$

$$W_{\text{REMOVED}}(t) \ge \left\lfloor \sum_{i=1}^{t} \frac{k}{4 \cdot c_i} \right\rfloor \ge \left\lfloor \frac{k \cdot t}{8|\partial F_0| - 8} \right\rfloor.$$

Thus, we are interested in

$$\left\lfloor \frac{k \cdot t_{\text{SUCCESS}}}{8(|\partial F_0| - 1)} \right\rfloor \ge W(F_0) + k.$$

Since releasing the agents requires an additional $2k$ time steps, the final bound for this case is

$$t_{\text{SUCCESS}} \ge \frac{8(|\partial F_0| - 1) \cdot (W(F_0) + k)}{k} + 2k.$$

If the t_{SUCCESS} obtained is greater than d, we continue to the next step. Lemma 6 provides us with c_t. As for $W_{\text{REMOVED}}(t_{\text{SUCCESS}})$, remembering that the actual cleaning begins at $t = 2 \cdot k$ (the time it takes to release the agents), we get

$$\begin{aligned} & W_{\text{REMOVED}}(t_{\text{SUCCESS}}) \\ & \ge \left\lfloor \sum_{i=2k}^{t_{\text{SUCCESS}}} \frac{k}{4 \cdot c_i} \right\rfloor \ge \left\lfloor \sum_{i=2k}^{d} \frac{k}{4 \cdot c_i} \right\rfloor + \left\lfloor \sum_{i=d+1}^{t_{\text{SUCCESS}}} \frac{k}{4 \cdot c_i} \right\rfloor \\ & \ge \left\lfloor \frac{(d-2k) \cdot k}{8(|\partial F_0| - 1)} \right\rfloor + \left\lfloor \sum_{i=d+1}^{t_{\text{SUCCESS}}} \frac{k}{8 \cdot (S_0 + \lfloor \frac{i}{d} \rfloor \cdot c_0 + 2(\lfloor \frac{i}{d} \rfloor^2 + \lfloor \frac{i}{d} \rfloor) - 1)} \right\rfloor. \end{aligned}$$

As we already know that $W(F_{i+1}) \le W(F_0) + \lfloor \frac{i}{d} \rfloor$, using the previous expression for W_{REMOVED} yields

$$\left\lfloor \sum_{i=2k}^{d} \frac{k}{4 \cdot c_i} \right\rfloor + \left\lfloor \sum_{i=d+1}^{t} \frac{k}{4 \cdot c_i} \right\rfloor \ge W(F_0) + k + \left\lfloor \frac{t}{d} \right\rfloor$$

$$\Longrightarrow \quad \left\lfloor \sum_{i=d+1}^{t} \frac{k}{4 \cdot c_i} \right\rfloor \ge W(F_0) + k + \left\lfloor \frac{t}{d} \right\rfloor - \frac{(d-2k) \cdot k}{8(|\partial F_0| - 1)}$$

$$\Longrightarrow \quad \sum_{i=d+1}^{t} \frac{1}{S_0 - 1 + 2\lfloor \frac{i}{d} \rfloor^2 + (c_0 + 2)\lfloor \frac{i}{d} \rfloor} \ge \left(\frac{8(k + W(F_0))}{k} - \frac{d-2k}{|\partial F_0| - 1} \right) + \frac{8}{k} \cdot \left\lfloor \frac{t}{d} \right\rfloor.$$

In order to find a time t which satisfies this expression, one should add elements to the sum one by one, recalculating the value of the right-hand expression. The minimal t for which the above inequality holds is t_{SUCCESS}. Performing the above takes $O(t_{\text{SUCCESS}})$ time (since every iteration requires a single comparison). □

Let $t_{\text{NoSpread}}(k)$ denote the time a region can be cleaned in using the **SWEEP** protocol, fast enough such that no spreads can take place. Notice that the expression

$$t_{\text{NoSpread}}(k) \triangleq \frac{8(|\partial F_0| - 1) \cdot (W(F_0) + k)}{k} + 2k$$

is not monotonic. Moreover, we can see that

$$\lim_{k \to \infty} t_{\text{NoSpread}}(k) = \infty.$$

Note that given k agents, using merely a portion of them is naturally a valid strategy, if better results result due to this. Therefore, we can state that the following corollary.

Corollary 1. *Given a contaminated region F_0 and k cleaning agents using the **SWEEP** protocol, let us define*

$$\hat{k} \triangleq \sqrt{4 \cdot W(F_0) \cdot (|\partial F_0| - 1)}$$

and let $\hat{t}_{\text{NoSpread}}$ be defined as

$$\min \left\{ \begin{array}{l} \frac{8(|\partial F_0| - 1) \cdot (W(F_0) + \lceil \hat{k} \rceil)}{\lceil \hat{k} \rceil} + 2\lceil \hat{k} \rceil, \\ \\ \frac{8(|\partial F_0| - 1) \cdot (W(F_0) + \lfloor \hat{k} \rfloor)}{\lfloor \hat{k} \rfloor} + 2\lfloor \hat{k} \rfloor. \end{array} \right\}$$

If $\hat{k} < k$ and $\hat{t}_{\text{NoSpread}} \le d$, then the completion of the cleaning mission before a contamination spread is guaranteed, and $t_{\text{success}} = \lceil \hat{t}_{\text{NoSpread}} \rceil$.

Proof. We are interested in the "effective" number of agents $\hat{k}$, which minimizes the value of t_{NoSpread}. Namely, a value $\hat{k}$ for which $\frac{dt_{\text{NoSpread}}}{dk}(\hat{k}) = 0$, provided that $\hat{k} \le k$. □

The bounds of Theorem 4 are illustrated in Figures 28 and 29.

6.4. Direct Upper Bounds

The bound in Theorem 4 contains a recursive expression and requires an iterative process in order to calculate the minimal time t_{SUCCESS} in which it is guaranteed that the agents will complete the cleaning mission. The following theorem presents an upper bound for the cleaning time given in a more readily compatible expression, whose solution is the requested value for t_{SUCCESS}.

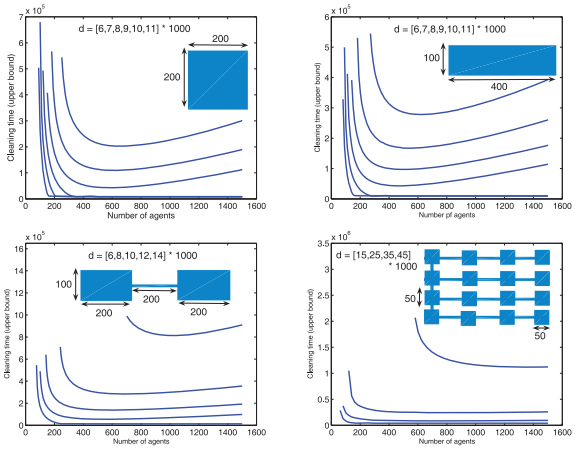

Fig. 28. Theorem 4 illustrated. Upper bound over the cleaning time is shown as a function of the number of cleaning agents using the **SWEEP** protocol. All four contaminated regions shown are of the same initial size.

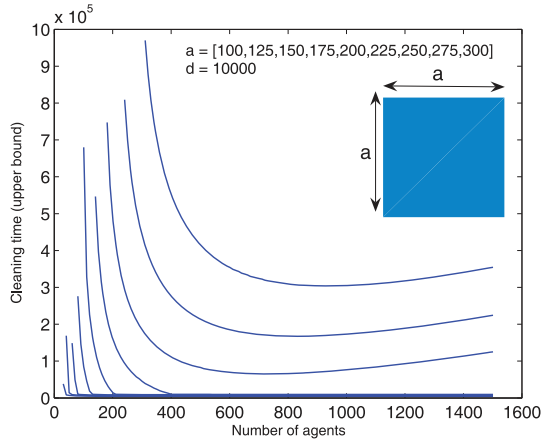

Fig. 29. Theorem 4 illustrated. Upper bound over the cleaning time is shown as a function of the number of cleaning agents using the **SWEEP** protocol, for $d = 10000$ and size of initial region varying between $S_0 = 10000$ and $S_0 = 90000$ (all other geometric properties remaining the same).

Theorem 5. *If* $(t = \frac{8(|\partial F_0|-1)\cdot(W(F_0)+k)}{k} + 2k)$ *is not greater than* d*, then* $t_{\text{SUCCESS}} = \lceil t \rceil$*. This also holds for static environments, since in such cases we simply have* $d \to \infty$*.*

Otherwise $(t > d)$*: find the minimal* μ *for which*

$$\psi\left(\mu+\gamma_{3_-}\right)-\psi\left(\mu+\gamma_{3_+}\right)+\gamma_1-\frac{\gamma_2\cdot\gamma}{d}-\frac{8\cdot\gamma_2}{d\cdot k}\cdot\mu\geq 0,$$

where

$$\gamma_1 \triangleq \psi\left(1+\gamma_{3_+}\right)-\psi\left(1+\gamma_{3_-}\right),$$

$$\gamma_{3_-} \triangleq \frac{c_0+2-\gamma_2}{4}, \qquad \gamma_{3_+} \triangleq \frac{c_0+2+\gamma_2}{4},$$

$$\gamma_2 \triangleq \sqrt{(c_0+2)^2-8\cdot(S_0-1)}$$

and

$$\gamma \triangleq \frac{8(k+W(F_0))}{k}-\frac{d-2k}{|\partial F_0|-1},$$

where $\psi(\,)$ *is the* digamma *function (defined in Abramowitz and Stegun (1964))—the logarithmic derivative of the gamma* function*, defined as*

$$\psi(x)=\frac{d}{dx}\ln\Gamma(x)=\frac{\Gamma'(x)}{\Gamma(x)}$$

or as

$$\psi(x)=\int_0^\infty\left(\frac{e^{-t}}{t}-\frac{e^{-xt}}{1-e^{-t}}\right)dt.$$

Then $t_{\text{SUCCESS}} = \lceil \mu \cdot d \rceil$.

Proof. The first part of the theorem is equivalent to the first part of Theorem 4.

For the second part, recall that in the second part of Theorem 4 we were interested in finding the minimal t for which the following inequality holds:

$$\sum_{i=d+1}^{t}\frac{1}{S_0-1+2\lfloor\frac{i}{d}\rfloor^2+(c_0+2)\lfloor\frac{i}{d}\rfloor}\geq\gamma+\frac{8}{k}\cdot\left\lfloor\frac{t}{d}\right\rfloor,$$

where

$$\gamma \triangleq \frac{8(k+W(F_0))}{k}-\frac{d-2k}{|\partial F_0|-1}.$$

Let us denote by $\mu \triangleq \lfloor\frac{t}{d}\rfloor$ the number of spreads that had taken place. Therefore, we can now search for the minimal μ for which

$$\sum_{i=1}^{\mu}\frac{d}{S_0-1+2\cdot i^2+(c_0+2)\cdot i}\geq\gamma+\frac{8}{k}\cdot\mu.$$

Hence,

$$d\cdot\sum_{i=1}^{\mu}\frac{1}{S_0-1+2\cdot i^2+(c_0+2)\cdot i}\geq\gamma+\frac{8}{k}\cdot\mu.$$

The left-hand side of the inequality is in the form of

$$d\cdot\sum_{i=1}^{\mu}\frac{1}{A+B\cdot i+C\cdot i^2}, \tag{4}$$

where $A = S_0 - 1, \quad B = c_0 + 2, \quad C = 2.$

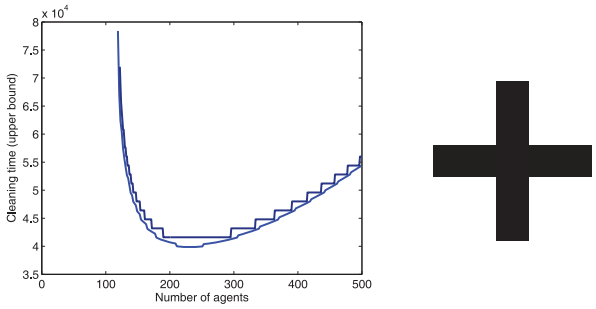

Fig. 30. An example of Theorem 5. The lower curve represents a bound over the cleaning time, as produced by Theorem 4, for the cross region which appears on the right (for which $S_0 = 2960$, $c_0 = 326$, $d = 1600$). The higher curve is produced by using Theorem 5.

This expression can be integrated into the form of

$$d \cdot \frac{\psi\left(\frac{2 \cdot C \cdot x + B - \sqrt{B^2 - 4 \cdot C \cdot A}}{2 \cdot C}\right) - \psi\left(\frac{2 \cdot C \cdot x + B + \sqrt{B^2 - 4 \cdot C \cdot A}}{2 \cdot C}\right)}{\sqrt{B^2 - 4 \cdot C \cdot A}} \Bigg|_1^{\mu}. \quad (5)$$

After assigning the proper values of A, B and C, we obtain the following formula, where the minimal μ which satisfies it will be μ_{SUCCESS}:

$$\frac{d}{\gamma_2} \cdot \left(\psi(\mu + \gamma_{3_-}) - \psi(\mu + \gamma_{3_+})\right) + \frac{d}{\gamma_2} \cdot \gamma_1 \geq \gamma + \frac{8}{k} \cdot \mu.$$

Here we have

$$\gamma_1 \triangleq \psi\left(1 + \gamma_{3_+}\right) - \psi\left(1 + \gamma_{3_-}\right),$$

$$\gamma_2 \triangleq \sqrt{(c_0 + 2)^2 - 8 \cdot (S_0 - 1)},$$

$$\gamma_{3_-} \triangleq \frac{c_0 + 2 - \gamma_2}{4}, \qquad \gamma_{3_+} \triangleq \frac{c_0 + 2 + \gamma_2}{4}$$

and

$$\gamma \triangleq \left(\frac{8(k + W(F_0))}{k} - \frac{d - 2k}{|\partial F_0| - 1}\right).$$

By rewriting the above inequality, we get

$$\psi(\mu + \gamma_{3_-}) - \psi(\mu + \gamma_{3_+}) + \gamma_1 - \frac{\gamma_2 \cdot \gamma}{d} - \frac{8 \cdot \gamma_2}{d \cdot k} \cdot \mu \geq 0.$$

Since $\mu = \lfloor \frac{t}{d} \rfloor$, $\mu > \frac{t}{d} + 1$ and therefore

$$t_{\text{SUCCESS}} < (\mu_{\text{SUCCESS}} - 1) \cdot d.$$

□

Figure 30 shows an example of applying Theorem 5 to the case of a cross-shaped region.

6.5. *Simplifying Theorem 5*

Although Theorem 5 presents a more direct way of calculating an upper bound over the cleaning time of k agents using the **SWEEP** protocol, this expression is still implicit. Theorem 6 presents an explicit approximation of the result of Theorem 5.

Theorem 6. *Let F_0 be a contaminated region for which Theorem 5 predicts that the contamination will spread before being cleaned. Then let* $\mu_{\text{SUCCESS}} \triangleq \min\{x \in \{\mu_1, \mu_2\} | x > 0\}$, *where μ_1 and μ_2 equal*

$$\frac{A_4 - A_1 A_3 \pm \sqrt{(A_1 A_3 - A_4)^2 - 4A_3(A_2 - A_1 - A_1 A_4)}}{2A_3},$$

where

$$A_1 = \frac{c_0 + 2 - \gamma_2}{4}, \qquad A_2 = \frac{c_0 + 2 + \gamma_2}{4},$$

$$A_3 = \frac{8 \cdot \gamma_2}{d \cdot k}, \qquad A_4 = \gamma_1 - \frac{\gamma_2 \cdot \gamma}{d}$$

and where

$$\gamma_1 \triangleq \psi\left(1 + A_2\right) - \psi\left(1 + A_1\right),$$

$$\gamma_2 \triangleq \sqrt{(c_0 + 2)^2 - 8 \cdot (S_0 - 1)}$$

and

$$\gamma \triangleq \frac{8(k + W(F_0))}{k} - \frac{d - 2k}{|\partial F_0| - 1}.$$

If such a μ_{SUCCESS} exists, then

$$t_{\text{SUCCESS}} = (\mu_{\text{SUCCESS}} - 1) \cdot d.$$

Proof. From Theorem 5, we know that if μ_{SUCCESS} is the minimal μ for which the following expression is greater or equal to zero:

$$\psi\left(\mu + \frac{c_0 + 2 - \gamma_2}{4}\right) - \psi\left(\mu + \frac{c_0 + 2 + \gamma_2}{4}\right) + \gamma_1 - \frac{\gamma_2 \cdot \gamma}{d} - \frac{8 \cdot \gamma_2}{d \cdot k} \cdot \mu$$

(for some constant γ, γ_1 and γ_2), then

$$t_{\text{SUCCESS}} = (\mu_{\text{SUCCESS}} - 1) \cdot d.$$

Note that this expression is in the form of

$$\psi\left(\mu + A_1\right) - \psi\left(\mu + A_2\right) - A_3 \cdot \mu + A_4 \geq 0. \quad (6)$$

As to the *digamma* function, we know that (consult Equation (6.3.5) in Abramowitz and Stegun (1964))

$$\psi(z + 1) = \psi(z) + \frac{1}{z}.$$

Thus, we can see that

$$\frac{A_1 - A_2}{\mu + A_1} \leq \psi\left(\mu + A_1\right) - \psi\left(\mu + A_2\right) \leq \frac{A_1 - A_2}{\mu + A_2}. \quad (7)$$

Combining inequalities (6) and (7), we see that we must determine the minimal μ for which

$$\frac{A_1 - A_2}{\mu + A_1} \geq A_3 \cdot \mu - A_4. \quad (8)$$

Hence, we find the minimal μ for which

$$A_3 \cdot \mu^2 + (A_1 A_3 - A_4) \cdot \mu + (A_2 - A_1 - A_1 A_4) \leq 0. \quad (9)$$

Note that $A_3 > 0$ and that $\mu \in (0, \infty)$. Thus, both roots have the same sign (where for negative roots there is no valid solution, and for positive roots the minimum of them is the solution). Roots of opposite sign are not possible, since this would imply the solution $\mu = 0$, which in turn is impossible as $F_0 \neq \emptyset$, and as at least one contamination spread took place. μ_1 and μ_2 are defined as

$$\frac{A_4 - A_1 A_3 \pm \sqrt{(A_1 A_3 - A_4)^2 - 4A_3(A_2 - A_1 - A_1 A_4)}}{2A_3}.$$

After some algebra, and by assigning the values of A_1, A_2, A_3 and A_4, we get the requested result. □

6.6. An Upper Bound for k

Using the previous upper bound for the time it takes k agents to clean an initially contaminated region of known geometric properties, a corresponding bound for the number of agents required to guarantee a successful cleaning of the region (in a finite amount of time) can be produced.

Theorem 7. *Given a contaminated region of properties S_0, $|\partial F_0|$ and $W(F_0)$, spreading every d time steps, in order to guarantee a successful cleaning of the region before the contamination is able to spread even once, the minimal number of cleaning agents required is upper bounded as follows:*

$$(k_1 \leq k \leq k_2) \wedge (k > 0),$$

where

$$k_{1,2} = 2\delta \pm 2\sqrt{\delta^2 - (|\partial F_0| - 1)\, W(F_0)},$$

where

$$\delta = -|\partial F_0| + 1 + \frac{d}{8}.$$

Proof. Let us use the first part of Theorem 5:

$$\frac{8(|\partial F_0| - 1) \cdot (W(F_0) + k)}{k} + 2k \leq d.$$

After some algebra, we obtain

$$2k^2 + (8|\partial F_0| - 8 - d)\, k + (8|\partial F_0| - 8)\, W(F_0) \leq 0.$$

And, after extracting the value of k from the quadratic equation, the result follows. Theorem 7 is illustrated in Figures 31 and 32. □

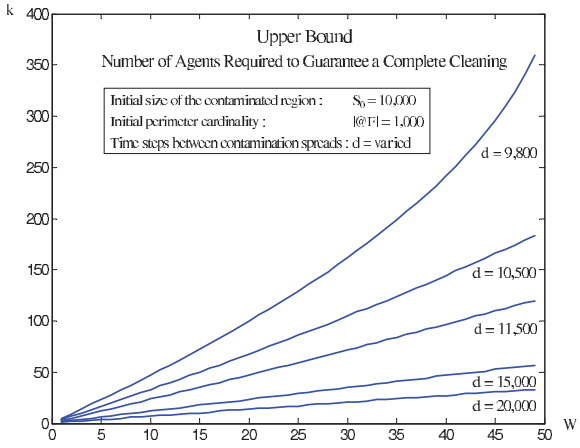

Fig. 31. An example of Theorem 7. The graph presents an upper bound of the number of agents needed to guarantee a successful cleaning of a region of initial size of 10,000 tiles and a perimeter cardinality of 1,000 tiles, as a function of the width of the region, namely—its "bulkiness". Notice how the effect this geometric feature has on the number of agents required increases dramatically as the time intervals between contamination spreads decrease.

It would be interesting to investigate whether a similar result can be produced, in the form of a lower bound for the number of agents required for cleaning a region while at most m spreads take place. Such a result can be derived from Theorems 5 and 6, and is the topic of another paper that is currently in preparation.

7. Related Work

As mentioned in previous sections, the cooperative cleaners problem has significant similarity to other types of multi-agent problems, such as cooperative coverage or cooperative demining problems. In recent years, a lot of effort went into designing systems and algorithms for handling such tasks. In this process, various models and assumptions concerning the agents and their capabilities were used.

In general, most of the techniques used for the task of a distributed coverage use some sort of cellular decomposition. For example, in Rekleitis et al. (2004) the area to be covered is divided between the agents based on their relative locations. In Butler et al. (2001) a different decomposition method is being used, which is analytically shown to guarantee a complete coverage of the area. Another interesting work is presented in Acar et al. (2001), discussing two methods for cooperative coverage (one probabilistic and the other based on an exact cellular decomposition). All of the works mentioned above, however, rely on the assumption that the cellular decomposition of the area is possible. This, in turn, requires the use of memory resources, used for storing the dynamic map generated, the boundaries of the cells, etc. As the initial

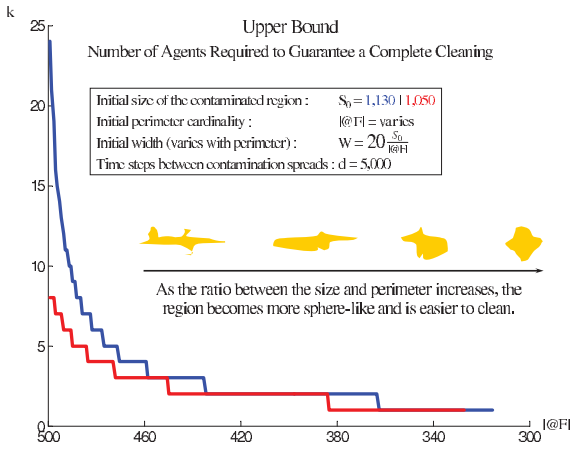

Fig. 32. An example of Theorem 7. Observe how as $\frac{S_0}{\partial F_0}$ increases (namely, the initial region is "bulkier"), it becomes easier to guarantee the proper cleaning of the region with a smaller number of agents. The width of the initial region was selected to reflect a reasonable proportion considering the changing cardinality value of the perimeter ∂F_0.

size and geometric features of the area are generally not assumed to be known in advance, agents equipped with merely a constant amount of memory will not be able to use such algorithms. In this work, we were interested in a multi-agent system which could perform cooperative coverage with use of a minimal amount of memory, unrelated to the size and geometry of the covered (or cleaned) area (this requirement is presented in Section 3). Such a system was analyzed in the later part of this work, and its ability to guarantee a completion of the task was shown.

Surprisingly, while some existing works concerning distributed (and decentralized) coverage present analytic proofs for the ability of the system to guarantee the completion of the task (for example, in Acar et al. (2001), Butler et al. (2001) and Batalin and Sukhatme (2002)), most of them lack analytic bounds for the coverage time (although in many cases an extensive amount of empirical results of this nature is made available by extensive simulations). Although a proof for the coverage completion is an essential element in the design of a multi-agent system, analytic indicators for its efficiency are in our opinion of great importance. We provide some such results, as bounds for the cleaning time of the agents, in Sections 4,5 and 6 of this work.

An interesting work to mention in this context is that of Koenig and collaborators (Koenig and Liu 2001; Svennebring and Koenig 2004), where a swarm of ant-like robots is used for repeatedly covering an unknown area, using a real-time search method called *node counting*. By using this method, the robots are shown to be able to efficiently perform such a coverage mission, and analytic bounds for the coverage time are discussed. Another work discussing a decentralized coverage of terrains is presented in Zheng and Koenig (2007). This work examines domains with non-uniform traversability. Completion times are given for the proposed algorithm, which is a generalization of the forest search algorithm. In this work, though, the region to be searched is assumed to be known in advance—a crucial assumption for the search algorithm, which relies on a cell-decomposition procedure.

Vertex-ant-walk, a variation of the node-counting algorithm presented in Wagner et al. (1998), is shown to achieve a coverage time of $O(n\delta_G)$, where δ_G is the graph's diameter. This work is based on a previous work in which a coverage time of $O(n^2\delta_G)$ was demonstrated (Thrun. 1992). Another work, called *Exploration as graph construction*, provides a coverage of degree-bounded graphs in $O(n^2)$ time, and can be found in Dudek et al. (1991). Here a group of limited ant robots explores an unknown graph using special "markers".

As most coverage problems focus on achieving complete coverage (and sometimes in minimal time), it is worth mentioning in this scope the work of Conner et al. (2005), in which an interesting additional constraint is added. As this work was designed for an autonomous painting system, the uniformity of the coverage was also demanded (in order to maintain the same thickness of the paint layer). In addition, this work examined the problem of 3-D coverage, contrary to most search and cover methods focusing on 2-D versions. We note nevertheless that the main focus of this work was the use of an efficient single agent for the mission, instead of dealing with a cooperative multi-agent setting.

The cooperative cleaners problem is also strongly related to the problem of distributed search after mobile and evading target(s) (Koenig et al. 2007; Altshuler et al. 2008; Borie et al. 2009) or the problems discussed under the names of "Cops and robbers" or "Lions and men" pursuits (Isler et al. 2006).

8. Experimental Results

Several simulation implementations of the **SWEEP** cleaning protocol were made (these simulated environments are currently available online at Technion's Intelligent Systems Laboratory's web site). We note that the full implementation of the **SWEEP** protocol involved writing code that effectively implemented a finite data automaton agent with $O(1)$ local memory, working according to the rules above. A multi-agent system was then simply a setting where at each initial location such an agent automaton was instantiated and then the system simply ran its course of cleaning the environment.

Exhaustive simulations were carried out, examining the cleaning activity of the agents in regions with various geometric features. Figures 33, 34 and 35 depict the cleaning process of several contaminated regions—for each one, the cleaning time for various numbers of agents is presented. As the agents' cleaning times are influenced by their initial locations, for every number of agents, a multitude of simulations were performed, one for each

possible location (namely, the total number of simulations for any number of agents equaled $|\partial F_0|$). The maximal and minimal cleaning times for each number of agents are measured, as well as the average cleaning time (calculated for all initial locations). Figure 36 for example demonstrates how placing the agents at different starting points allows them to successfully clean regions which were impossible to clean when the agents were concentrated at a single starting point. Figure 37 demonstrates the dynamic nature of the agents' cleaning efficiency, by comparing the contaminated region's size to the whereabouts of the agents at various times. Figure 38 demonstrates the changes in the geometric features of a region while it is being cleaned, as well as the influence the number of agents has on the efficiency of the cleaning.

9. Discussion and Conclusion

This work described the *dynamic cooperative cleaners* problem, where a group of simple and limited agents must clean a dynamic "contaminated" region spreading due to contamination of neighbors. This problem has several interesting applications, some of which are the coordination of fire-fighting units (see Casbeer et al. (2006) for details) or an implementation of a distributed anti-virus system for computer networks (Janakiraman et al. 2003). Additional applications are distributed search engines (Bender et al. 2005), as well as various military applications. The problem is also related to the geometric problem of pocket machining (Held 1991). An interesting problem of cleaning and maintenance of a system of pipes by an autonomous agent is discussed in Neubauer (1993) and a multi-agent version can be of much interest. The importance of cleaning hazardous waste by agents is described in Hedberg (1995).

As mentioned previously, our cooperative algorithms approach can be considered as a case of social behavior in the sense of Shoham and Tennenholtz (1995), where one induces multi-agent cooperation by forcing the agents to obey some simple "social behavior" guidelines.

A theoretical lower bound for the cleaning time of a given contaminated region was shown, yielding a way to exhibit input scenarios which are impossible to clean. **SWEEP**, a cooperative cleaning protocol to be implemented by each agent, was presented and its performance was compared to the lower bound. The algorithm's performance was then analyzed, and several analytic upper bounds on its completion time were discussed.

9.1. Protocol's Robustness

The issue of an algorithm's robustness and fault tolerance is a major aspect of robotic systems, as the environments in which the agents operate may often include unpredicted change from the original specification of the system, such as various kinds of noises, robots' malfunctions, etc.

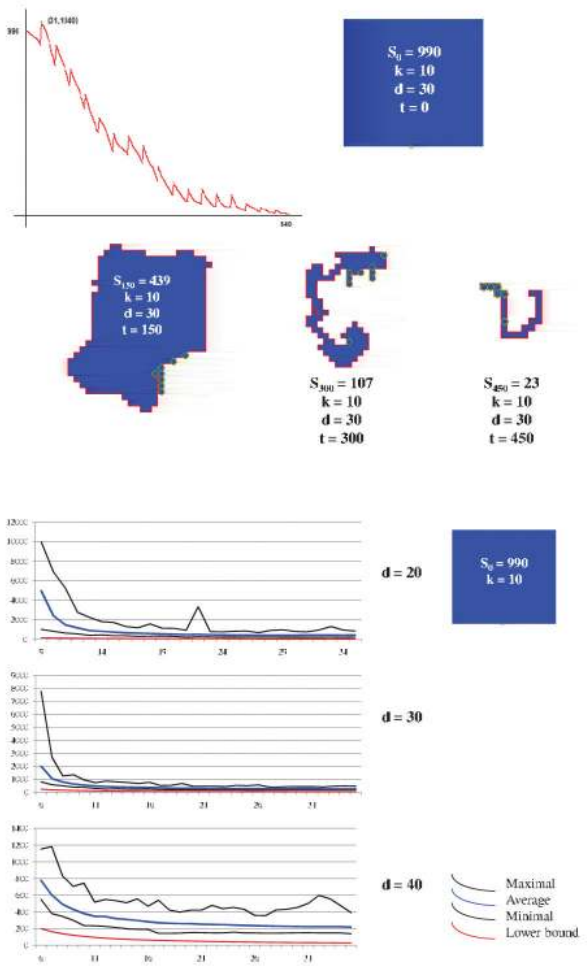

Fig. 33. An example of a cleaning mission by agents using the **SWEEP** cleaning protocol, trying to clean a contaminated square (see the original contaminated shape at the top, on the right). A graph of the size of the region as a function of the time, for 10 agents and a spreading delay of $d = 30$, appears in the left top corner, as well as size and shape of the region at several times during the cleaning. The three graphs at the bottom depict the cleaning time (maximal, minimal and average) for various numbers of agents, and for three values of spreading delay. In addition, the cleaning times are compared to the lower bound, as appears in Theorem 2.

While analyzing the performance of the cleaning protocol, the robotic agents were assumed to work perfectly. In real robotic systems, however, this is not always the case. In simulation, however, it was quite easy to examine what happens when one or more agents disappear. The disappearance of an agent which dies is necessary in our framework in order for the system to work properly.

As to the **SWEEP** protocol, using the analytic bounds presented in Section 6 (and specifically Theorems 4, 5, 6 and 7), we have shown that as long as enough agents are still functioning properly, the cleaning mission can still be accomplished successfully, albeit more slowly. But

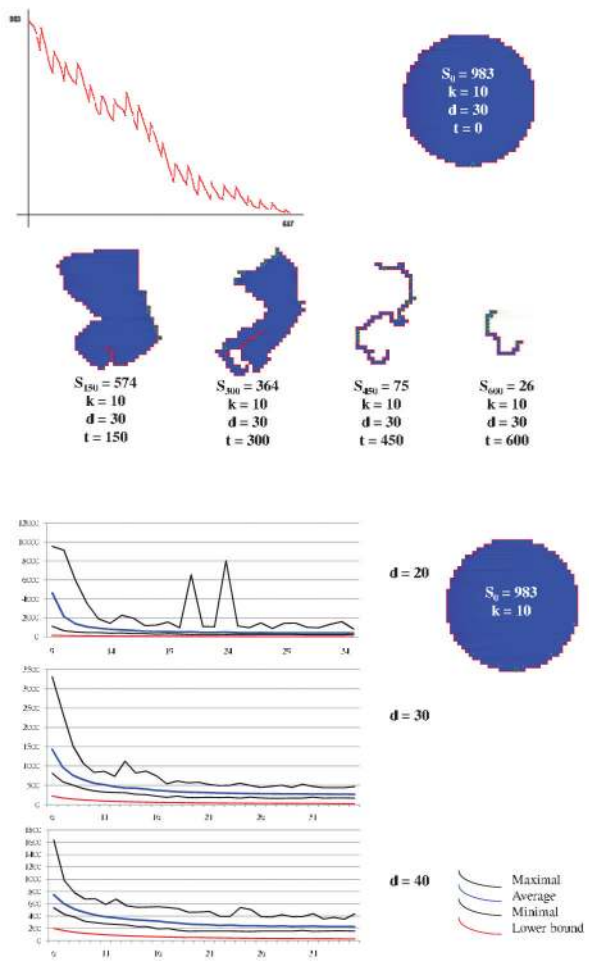

Fig. 34. An example of a cleaning mission by agents using the **SWEEP** cleaning protocol, trying to clean a contaminated circle (see the original contaminated shape at the top, on the right). A graph of the size of the region as a function of the time, for 10 agents and a spreading delay of $d = 30$, appears in the left top corner, as well as size and shape of the region at several times during the cleaning. The three graphs at the bottom depict the cleaning time (maximal, minimal and average) for various numbers of agents, and for three values of spreading delay. In addition, the cleaning times are compared to the lower bound, as appears in Theorem 2.

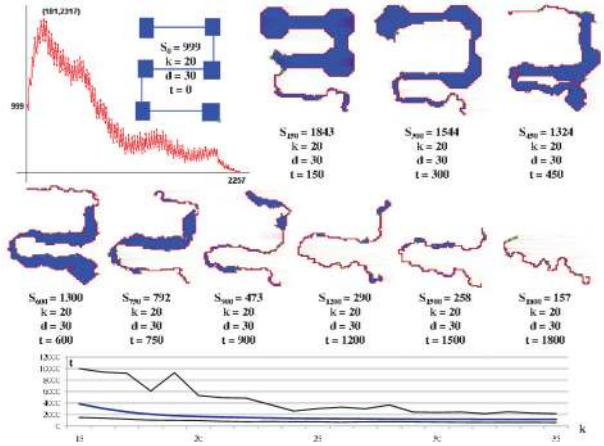

Fig. 35. An example of a cleaning mission by agents using the **SWEEP** cleaning protocol, trying to clean a contaminated region in the shape of several rooms, connected by narrow corridors (see the original contaminated shape at the top, on the right). A graph of the size of the region as a function of the time, for 20 agents and a spreading delay of $d = 30$, appears in the left top corner, as well as size and shape of the region at several times during the cleaning. The graph at the bottom depicts the cleaning time (maximal, minimal and average) for various numbers of agents, and for spreading delay of $d = 30$.

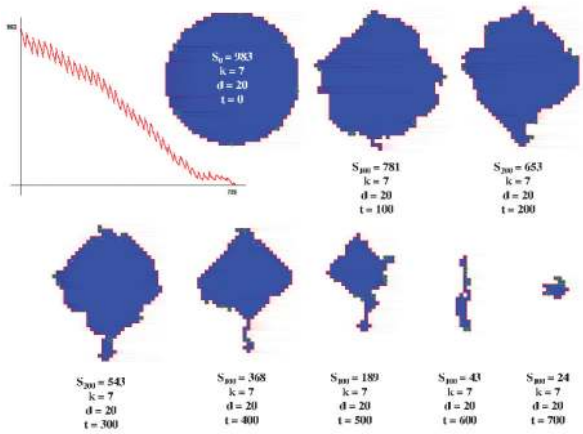

Fig. 36. The graph presents the evolution of the size and shape of a contaminated region in the shape of a circle, expanding every $d = 20$ time steps, being cleaned by $k = 7$ agents, using the **SWEEP** protocol).

what if some agents start to cheat? Such adversaries could have catastrophic consequences, since a malfunctioning or "crazy" agent may clean a critical point and disconnect the contaminated region.

As discussed in Section 5.3, one of the requirements of the **SWEEP** protocol is a complete fault tolerance to malfunctions in the agents, meaning that, even if one or several agents stop working ("die" and disappear) the rest of the agents will continue the cleaning process as efficiently as their number allows them. The basic idea enabling this robustness is the complete independence between the agents, due to the full decentralized nature of the system (as the agents do not share the cleaning mission between them, nor do they rely on one another in the performance of their cleaning process).

A different aspect of the algorithm's robustness is the performance of the agents in noisy environments. In general, random noise can appear and influence various aspects of the system—in sensing whether a tile should be cleaned, in sensing the presence of other agents in an agent's vicinity, in the agent's movement, etc. As in almost any system, it is easy to see that severe noise may significantly affect the algorithm's efficiency, and

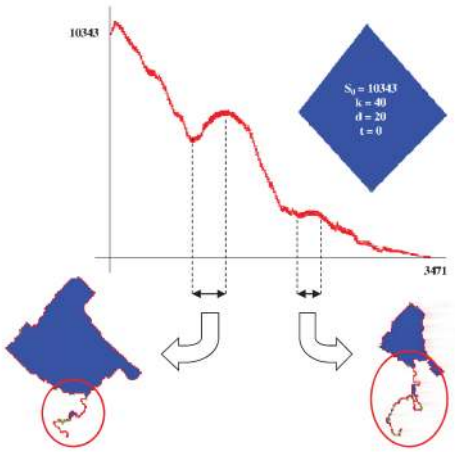

Fig. 37. The graph presents the cleaning process of a (relatively large) "digital sphere". Observe how the size of the region increases at various times. Examining the appearance of the region at those times reveals that the majority of agents are being located at the vicinity of the initial location of the agents (a point which is artificially kept critical, according to the **SWEEP** protocol)—see snapshots at the bottom. After several traversals of the region, this part of the region is composed mainly of critical points, which are not cleaned by the agents. Hence, when traveling through this part, the cleaning efficiency of the agents diminishes temporarily, and as the region continues to spread, its global size increases.

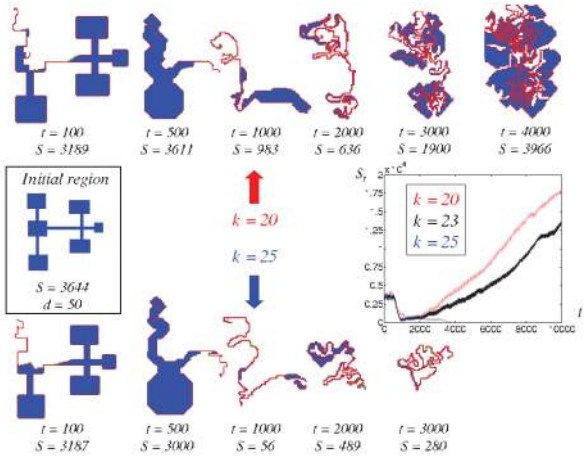

Fig. 38. This example demonstrates the change in the geometric properties of the contaminated region as a result of the cleaning process. Notice the "phase-transition" effect the number of agents has on the ability of the agents to entirely clean the region. Given three groups of agents, comprising 20, 23 and 25 agents, respectively, all three groups first achieve a very similar cleaning effect in the first 2000 time steps (presenting a significant decrease in the region's size), followed by a 1000 time step long plateau. However, whereas the first two group fail to continue cleaning the region, causing its size to grow significantly, the third group is able to complete the cleaning of the region using an additional 2000 time steps. Note that this dramatic change in the final state of the system is achieved after increasing the number of agents by less than 10%.

may even prevent the agents from completing the mission altogether. For example, difficulties in sensing whether a tile should be cleaned or not may cause an agent to clean a critical point, thus separating the region into several connected components. In addition, noises in the agents' movement system may cause the agents to detach from the contaminated region, thus delaying, or even preventing, the completion of the cleaning process. Whereas a situation in which the noise level is kept at a reasonable level (i.e. correct identification of clean and contaminated tiles, and movement noises), it is the authors' opinion that the completion of the algorithm may still be achieved, although the cleaning time in such a case is expected to increase. When the agent has difficulties in the sensors in charge of detecting other agents in its vicinity, and interpreting their signaling (see relevant sections for details about signaling and synchronization), several problematic scenarios (which these signaling mechanisms were designed to avoid) may occur. However, simulations show that in such cases often only the cleaning time is affected, while the completion of the cleaning mission is preserved. Nevertheless, as this issue was not fully investigated yet, a perfect neighbors' detection is still one of the algorithm's requirements.

One example for the above-described problem is presented in Figure 23, in which a failure in the sensing mechanism may damage the simple connectivity of the contaminated region. Note that in such scenarios, there is least a single agent in each of the new contaminated components. Therefore, each component is an instance of the cooperative cleaners problem, for which the upper bound for the cleaning time may be applied.

Another example is demonstrated in Figure 24, where a bug in the *waiting* mechanism (see Section 5.9 for more details) may prevent some of the agents from cleaning contaminated tiles. The same holds for possible bugs in the *resting* mechanism (see Section 5.10 for more details), which may cause several agents to move together, acting as a single cleaning agent.

9.2. Collisions and the Clustering Effect

Another question of interest is the resolution of collisions between agents—the formation of clusters of agents located in the same tile, and delaying each other. In this work we resolve such a problem by artificially producing an implicitly agreed order among the agents, depending on their previous locations and movements. However, it is an interesting open question whether randomization-based methods may achieve better results.

9.3. Probabilistic Cooperative Cleaners

While the methods proposed in the scope of this work, as well as the analysis methods used, were all deterministic, it

is interesting to examine the design of stochastic variants of these algorithms. It is our belief that such variants are expected to demonstrate increased robustness (mainly with regards to rogue robots, or various malfunctions and noises).

Furthermore, the model itself can be altered, to become more "realistic". For example, Definition 3 could be replaced with a probabilistic contamination expansion rule, according to which at every time step each contaminated tile has a (either fixed or environment-influenced) probability to contaminate its neighbors.

9.4. Agents' Communication

Another interesting point that may be interesting to discuss is that of inter-agent communication (i.e. the lack thereof). In this work we use the contamination on the floor as a means of implicit inter-agent communication. Still, whenever a close-range communication is required, other ways for communication between agents may be suggested. One is to use heat trails for this end, as was reported in Russell (1993). In Steels (1990), self organization is achieved among a group of lunar robots that have to explore an unknown region, and bring special rock samples back to the mother spaceship, by programming each robot to drop a pebble at each point he visits and walk around at random with a bias towards the negative gradient of the pebble's concentration.

9.5. Agents' Memory and Computation Resources

As mentioned previously, while implementing the **SWEEP** protocol the agents must use counters of $O(1)$ bits. Such counters are enough as we know (using the proof of Lemma 5 in Wagner et al. (2008)) that the maximal number of agents that may simultaneously reside in the same tile at any given moment is upper bounded by $O(1)$. Therefore, counting the agents located in the immediate vicinity can be done using counters of $O(1)$ bits. Note that this also implies that given a team of k agents cleaning an expanding region in the grid, towards the end of the cleaning process the agents are likely to be located at $O(\frac{k}{4})$ connected tiles, while each tile contains approximately four agents. When this scenario happens, it is identified by the *Check Near Completion of Mission* procedure of **SWEEP**, causing the agents to instantly clean the contaminated tiles. Notice that the execution of the *Check Near Completion of Mission* procedure requires each agent to communicate with $O(1)$ agents located in its immediate vicinity. This type of communication is allowed under our model's assumptions (similarly to the implementation of the *waiting* and *resting* mechanisms). In addition, the time it takes to execute the *Check Near Completion of Mission* procedure equals the time it takes to send $O(k)$ messages. We assume that the time required for processing a message is negligible compared to the time it takes the agents to move and clean tiles. Therefore, the execution time of the *Check Near Completion of Mission* procedure is still $O(1)$.

9.6. The "Elastic Barrier" Assumption

Throughout this paper two alternatives for the spreading of the contamination were used. A basic spreading model was first defined in Definition 4 in Section 3, assuming the connection between each two contaminated tiles. This basic model is sufficiently strong for the discussion of lower bounds and impossibility results, as appear in this work.

Later on, a second spreading model was introduced, simulating an "elastic barrier" which preserves the simple connectivity of the region throughout its evolution (Definition 9 in Section 5.4). Such an assumption was required to alow for the proper analysis of the **SWEEP** protocol. While discussing the need for this assumption, it should first be noted that the barrier assumption is not mandatory for the execution of the protocol, but only for its analysis. Namely, even if holes are created (and provided the agents are not "trapped" inside them), the protocol would function properly, albeit less efficiently. Furthermore, even if no barrier is assumed, our extensive simulation experiments show that for many initial contaminated regions the performance of the protocol will not be affected at all.

The analysis of the agents' performance under the "basic" model, with no barrier, is still an open problem. We believe that this can be done by further analyzing the effect of the spread of contamination which is not confined by a barrier on the geometrical properties of the contaminated region.

We believe that, in spite of the fact that the elastic barrier is non-physical and requires a weakening of the behavior of the dynamic world, the agents assumed in this work retain their very limited capabilities and the problem remains an example of a truly distributed swarm of identical agents, competing with an adversary which has a well-defined and local rule of changing the environment.

9.7. Dynamic Cooperative Cleaning Feasibility

An important further aspect of the problems investigated herein is the feasibility question, namely, estimating the minimal number of agents required to guarantee a proper cleaning of a given contaminated region (regardless of the cleaning time). From the upper bounds presented in Theorems 4, 5 and 6, some rough estimates can be derived. However, it seems that additional investigation of this subject is necessary.

An interesting example of the difficulty of producing accurate estimates for the cleaning time of a contaminated region is presented in Figure 39. This example (in which the size of the contaminated region increases considerably before the drop towards a complete cleaning of the region)

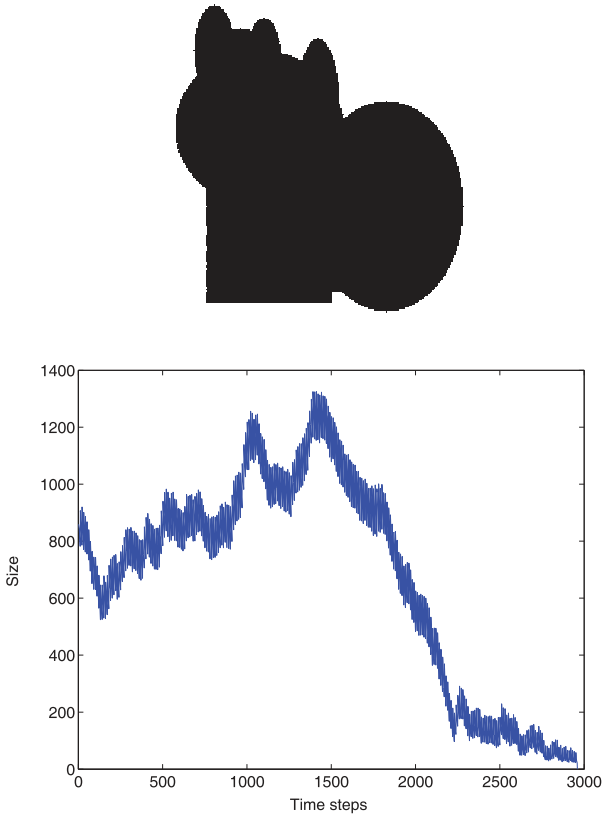

Fig. 39. This example demonstrates the difficulty of producing accurate estimations for the cleaning time of contaminated regions. The initial contaminated region appears as the left-hand chart whereas the graph on the right presents the size of the region as the cleaning process of agents using the **SWEEP** protocol advances. Notice that for about two-thirds of the cleaning time, the size of the contaminated region is actually greater than that of the initial region. Observing this graph, we can see that during the first half of the cleaning process, the geometric features of the contaminated region undergo dramatic changes, resulting in a region of the same size, but different in its other features (which apparently make it easier for the agents to clean it). The importance of this example is its demonstration of a counterexample for any claim that a tight upper bound for the cleaning time which takes advantage of the initial size of the contaminated region is possible.

demonstrates why it is impossible to generate tighter upper bounds for this problem, using merely the initial size of the contaminated region.

It is of interest to notice here that an off-line version of the problem, that is: finding the shortest path that visits all grid points in F, where F is completely known in advance, is *NP-hard* even for a single agent. It is a direct consequence of the fact that the Hamilton path in a non-simple grid graph is *NP-complete* (Itai et al. 1982).

9.8. Cleaning Style

Our approach presented in this work is that cleaning is always done at the boundary, in a "layers peeling" fashion. However, it is possible that better efficiency would be achieved using other approaches:

1. Given that several neighbors are contaminated, visit the non-critical ones first (even if not on the boundary). This approach is quite efficient for one agent, but its effect when multiple agents are involved is hard to anticipate, as well as the deadlock that may result.
2. When entering a large "room" (that is—upon passing from a critical area to a non-critical one)—we could designate the entrance by a special type of token, so that other agents will enter only in the case there is no other work to do. This approach may guarantee that the agents will be distributed between the large rooms of the F configuration. This is attractive if the region has such rooms of quite similar areas. The use of tokens of limited data capacity is similar to the use of *pheromones* by many insects. As was mentioned previously, studies of ants (e.g. Adler and Gordon (1992))show that the pheromone-based search strategies used by ants in foraging for food in unknown terrains tend to be very efficient. It is believed that ants build a network of information with vertices represented by points of encounter between ants' agents and the information is passed either between ants at a vertex with a physical encounter with other ants or via pheromone traces that are left on the ground. Inspired by nature, an interesting model for communicating in multi-agent systems is that of ant-walk (e.g. Yanovski et al. (2001)). In this model, information is spread to other agents via small amounts of data that are written by an agent at various places in the environment (e.g. edges or vertices in the graph) and can be later used or modified by other agents visiting that vertex. For additional reading on such approaches, see also Felner et al. (2006).
3. Quite a different idea is to divide the work into two phases. In the first phase (the "distribution phase"), the agents locate themselves uniformly around the area. Then, in the second phase, each agent cleans around his "center". If the distribution is appropriate, the number of interactions between agents in the second phase (which is a major cause for decreasing the cleaning efficiency) should be minimal. The problem of calculating and forming the optimal distribution, however, is still to be solved.
4. Understanding that the efficiency of the agents' cleaning is significantly affected by the geometric features of the contaminated region, and that those features are dynamic (due to the contamination spreading and the agents' cleaning), a division of the agents into two groups could also be made, one in charge of actively controlling various geometric features of the contaminated region by cleaning or even contaminating specific tiles, while the second takes care of the actual cleaning itself. The idea behind this scheme is that although the number of agents in the "cleaning group" is smaller than the total number of

agents available, the improved efficiency of those agents will compensate their decreased number, resulting in an overall improvement of the swarm's cleaning performance. One such geometric feature might be the ratio between the region's perimeter and its size (also known as the *shape factor*: $\frac{|\partial F_t|}{S_t}$). For example, it can be seen that if the shape factor is guaranteed to be bounded throughout the entire cleaning process as follows: $\forall t, \quad \frac{|\partial F_t|}{S_t} \geq \Upsilon$, tighter analytic bounds can be produced (Altshuler et al. 2006a).

In closing, we would like to quote a statement made by H.A. Simon after watching an ant making his laborious way across a wind-and-wave-molded beach (Simon 1981):

> "An ant, viewed as a behaving system, is quite simple. The apparent complexity of its behavior over time is largely a reflection of the environment in which it finds itself."

Such a point of view, as well as the results of our simulations and analysis, shows that even simple, ant-like robotic creatures, when properly programmed, may lead to very interesting, adaptive and quite efficient emergent goal-oriented behaviors.

Appendix: On the Isoperimetric Inequality in Grid Domains

In this section we discuss the isoperimetric inequality in grid regions. Namely, that the digital shape of a given area which has the minimal number of neighbors is a digital sphere. This result was previously required by Theorem 1 in Section 4. For more general results on different types of discrete lattices, see Vainsencher and Bruckstein (2008).

Theorem 8. *For all the shapes of area S, let F be the shape with the minimal number of clean* 4*Neighbors; then the number of clean* 4*Neighbors of F is at least the number of* 4*Neighbors of the largest digital sphere of size at most S.*

Proof. We first need the following definition.

Definition 21. Let l_{ru} and l_{ld} denote two infinite sets of tiles, each organized as a straight line slanted by 45 degrees (in the lower right direction). Similarly, let l_{lu} and l_{rd} denote two infinite sets of tiles, each organized as a straight line slanted by 45 degrees (in the lower left direction).

For any shape F, let l_{ru} be placed above and to the right of F, let l_{lu} be placed above and to the left of F, let l_{rd} be placed below and to the right of F and let l_{ld} be placed below and to the left of F. Let us move l_{ru} and l_{rd} to the left, until they touch F. In addition, let us move l_{lu} and l_{ld} to the right, until they touch F. Note that the four lines form a bounding slanted rectangle around F.

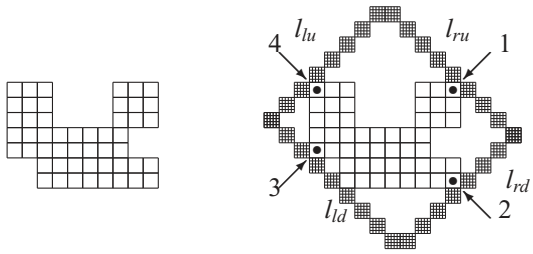

Fig. 40. An example of a *slanted-bounding-rectangle*.

Definition 22. Let us remove all the tiles of l_{ru}, l_{lu}, l_{rd} and l_{ld} which are not part of this rectangle, and denote the remaining tiles by *slanted-bounding-rectangle*(F).

Definition 23. For each of the tile sets l_{ru}, l_{rd}, l_{ld} and l_{lu}, let us denote the last tiles of F that are 4*Neighbors* of the sets (assuming clockwise movement) by 1, 2, 3 and 4, respectively.

An example appears in Figure 40.

Let us project all the tiles of slanted-bounding-rectangle(F) that are located between points 1 and 2 leftwards. Similarly, let us project the tiles between points 2 and 3 upwards, the tiles between point 3 and 4 rightwards and the tiles between points 4 and 1 downwards, until they all become *4Neighbors* of the tiles of F. It is easy to see that after this projection each tile of *slanted-bounding-rectangle*(F) will be "met" by at least a single tile of F. In addition, it is impossible that after the projection, two tiles of *slanted-bounding-rectangle*(F) will merge, since for some projecting direction, there is at most one tile of each projection coordinate (namely, when projecting left there is at most one tile for each Y coordinate, when projecting downwards there is at most one tile for each X coordinate, etc). Thus, the number of 4*Neighbors* of F is at least the number of tiles in *slanted-bounding-rectangle*(F), denoted by *slanted-bounding-rectangle*(F), namely:

$$\forall F, \quad 4Neighbors(F) \geq |\, slanted\text{-}bounding\text{-}rectangle(F)\,|. \tag{10}$$

Note that *slanted-bounding-rectangle*(F) in Figure 40 contains two pairs of tiles which are 4 connected.

Definition 24. For each slanted rectangle R, let us define *canonical-slanted-rectangle*(R) to be the smallest slanted rectangle which bounds R such that *canonical-slanted-rectangle*(R) does not contain pairs of tiles which are 4 connected. In other words, assuming that the world is a chessboard colored in black and white, then *canonical-slanted-rectangle*(R) is the minimal slanted rectangle that contains R, whose tiles have the same color.

It can easily be seen that for some slanted rectangle R, in order to produce

$$CR = canonical\text{-}slanted\text{-}rectangle(R),$$

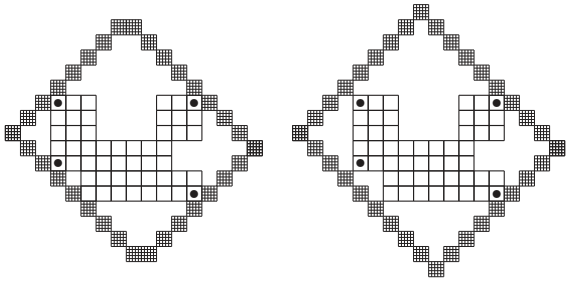

Fig. 41. For a shape F, the right-hand chart demonstrates *slanted-bounding-rectangle*(F) while the left-hand chart demonstrates *canonical-slanted-rectangle (slanted-bounding-rectangle*(F)*)*.

at most two sides of the four sides of R must be moved by one tile. In addition, each time a side of R is moved, its length should be increased by at most 1. As a result, the number of tiles in CR is at most the number of tiles in R plus 2, namely

$$\forall \textit{ rectangles } R, \quad |\textit{ canonical-slanted-rectangle}(R)| \leq |R| + 2. \tag{11}$$

An example appears in Figure 41.

Combining Equations (10) and (11), we see that for every region F, the value of $4\textit{Neighbors}(F)$ is greater than or equal to

$$|\textit{canonical-slanted-bounding-rectangle}(F)| - 2. \tag{12}$$

Let CR be the smallest canonical slanted rectangle which contains at least S tiles. Let a and b denote the sides of CR and let p denote the number of tiles CR comprises. Then

$$p = 2(a+b) - 4, \tag{13}$$

which also means that

$$a = \frac{p+4}{2} - b.$$

Let $f(a,b)$ denote the area of a canonical slanted rectangle of sides a and b. Since a canonical slanted rectangle contains $a-1$ slanted rows of $b-1$ tiles and $a-2$ slanted rows of $b-2$ tiles, we see that

$$f(a,b) = (a-1)(b-1) + (a-2)(b-2). \tag{14}$$

We would now like to find a solution for the following optimization problem:

$$\min \quad p \quad \text{s.t.} \quad f(a,b) \geq S \qquad \wedge \qquad p = 2(a+b) - 4.$$

After some arithmetic, Equation (14) can be written as

$$a = \frac{f(a,b) - 5 + 3b}{2b-3}. \tag{15}$$

Combining this with Equation (13), we get

$$p = 2 \cdot \frac{f(a,b) - 5 + 3b}{2b-3} + 2b - 4.$$

Since we require that $f(a,b) \geq S$, we can write the following:

$$p \geq \rho \triangleq 2 \cdot \frac{S - 5 + 3b}{2b-3} + 2b - 4. \tag{16}$$

Note that the value of b which will minimize ρ (denoted by $b_{\min}$) may not be an integer value, where the meaning of b is the length of the side of CR, which must be an integer number. However, since for all $b \in \mathbb{N}$, $\rho(b) \geq \rho(b_{\min})$ and since $p \geq \rho$, the validity of the bound is preserved (however, the bound may become slightly less tight). In order to minimize ρ, we shall calculate

$$\frac{\partial \rho}{\partial b} = 2 \cdot \frac{3(2b-3) - 2(S - 5 + 3b)}{(2b-3)^2} + 2 = 0$$

and, after some arithmetic, we get

$$b = \frac{\sqrt{2S-1} + 3}{2}.$$

By examining the behavior of $\frac{\partial^2 \rho}{\partial b^2}$, we can see that for $b = \frac{\sqrt{2S-1}+3}{2}$, since $S \geq 1$, then $\frac{\partial^2 \rho}{\partial b^2} > 0$, meaning that ρ is indeed minimized at this point.

By assigning the value of $b_{\min}$ to Equations (15) and (16), we can see that for $b_{\min}$, $a = b$ (meaning that CR is the shape of the perimeter of a digital sphere) and that

$$p \geq 2(\sqrt{2S-1} + 1).$$

It is easy to see that a sphere of size S has exactly $2(\sqrt{2S-1} + 1)$ $4\textit{Neighbors}$ and therefore it is the shape that minimizes the number of $4\textit{Neighbors}$ for a given area S.

Since the bound was produced for a *canonical* slanted rectangle, and combined with Equation (12), we get

$$\forall F \textit{ of size } S, \quad 4\textit{Neighbors}(F) \geq 2\sqrt{2S-1}.$$

□

Notes

1. For counting purposes, the agents must be equipped with counters that can store the number of agents in their immediate vicinity. This can of course be implemented using $O(\log k)$ memory. However, throughout the proof of Lemma 5 in Wagner et al. (2008), it is shown that the maximal number of agents that may simultaneously reside in the same tile at any given moment is upper bounded by $O(1)$. Therefore, counting the agents in the immediate vicinity can be done using counters of $O(1)$ bits.

References

Abramowitz, M. and Stegun, I. (1964). *Handbook of Mathematical Functions*, Vol. 55 (*Applied Mathematics Series*). National Bureau of Standards.

Acar, E., Zhang, Y., Choset, H., Schervish, M., Costa, A., Melamud, R., Lean, D. and Gravelin, A. (2001). Path planning for robotic demining and development of a test platform. *International Conference on Field and Service Robotics*, pp. 161–168.

Adler, F. and Gordon, D. (1992). Information collection and spread by networks of partolling agents. *The American Naturalist*, 140(3): 373–400.

Agmon, N., Sadov, V., Kaminka, G. and Kraus, S. (2008). The impact of adversarial knowledge on adversarial planning in perimeter patrol. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 55–62.

Agogino, A. and Tumer, K. (2008). Regulating air traffic flow with coupled agents. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 535–542.

Altshuler, Y., Wagner, I. and Bruckstein, A. (2006a). Shape factor's effect on a dynamic cleaners swarm. *Third International Conference on Informatics in Control, Automation and Robotics (ICINCO), the Second International Workshop on Multi-Agent Robotic Systems (MARS)*, pp. 13–21.

Altshuler, Y., Yanovski, V., Vainsencher, D., Wagner, I. and Bruckstein, A. (2006b). On minimal perimeter polyminoes. *The 13th International Conference on Discrete Geometry for Computer Imagery (DGCI2006)*, pp. 17–28.

Altshuler, Y., Yanovsky, V., Bruckstein, A. and Wagner, I. (2008). Efficient cooperative search of smart targets using UAV swarms. *ROBOTICA*, 26: 551–557.

Angluin, D., Aspnes, J. and Eisenstat, D. (2008). A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21: 87–102.

Arkin, R. and Balch, T. (1997). Aura: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2/3): 175–188.

Baeza-Yates, R. and Schott, R. (1995). Parallel searching in the plane. *Computational Geometry*, 5: 143–154.

Batalin, M. and Sukhatme, G. (2002). Spreading out: a local approach to multi-robot coverage. *6th International Symposium on Distributed Autonomous Robotics Systems*.

Bender, M., Michel, S., Triantafillou, P., Weikum, G. and Zimmer, C. (2005). Minerva: collaborative P2P search. *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 1263–1266.

Bendjilali, K., Belkhouche, F. and Belkhouche, B. (2009). Robot formation modelling and control based on the relative kinematics equations. *The International Journal of Robotics and Automation*, 24(1): 3220.

Bhatt, R., Tang, C. and Krovi, V. (2009). Formation optimization for a fleet of wheeled mobile robots — a geometric approach. *Robotics and Autonomous Systems*, 57(1): 102–120.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. New York, Oxford University Press.

Borie, R., Tovey, C. and Koenig, S. (2009). Algorithms and complexity results for pursuit–evasion problems. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 59–66.

Braitenberg, V. (1984). *Vehicles*. Cambridge, MA, MIT Press.

Brooks, R. (1990). Elephants don't play chess. *Designing Autonomous Agents*, pp. 3–15.

Brooks, R. and Flynn, A. (1989). Fast, cheap and out of control, a robot invasion of the solar system. *Journal of the British interplanetary Society*, 42: 478–485.

Bruckstein, A. (1993). Why the ant trails look so straight and nice. *The Mathematical Intelligencer*, 15(2): 59–62.

Bruckstein, A., Mallows, C. and Wagner, I. (1997). Probabilistic pursuits on the integer grid. *American Mathematical Monthly*, 104(4): 323–343.

Butler, Z., Rizzi, A. and Hollis, R. (2001). Distributed coverage of rectilinear environments. *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*.

Casbeer, D., Kingston, D., Beard, R. and McLain, T. (2006). Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science*, 37(6): 351–360.

Connaughton, R., Schermerhorn, P. and Scheutz, M. (2008). Physical parameter optimization in swarms of ultra-low complexity agents. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1631–1634.

Conner, D., Greenfield, A., Atkar, P., Rizzi, A. and Choset, H. (2005). Paint deposition modeling for trajectory planning on automotive surfaces. *IEEE Transactions on Automation Science and Engineering*,2(4): 381–392.

Dias, M. and Stentz, A. (2001). *A Market Approach to Multirobot Coordination*. Technical Report CMU-RI-TR-01-26, Robotics Institute, Carnegie Mellon University.

Dudek, G., Jenkin, M., Milios, E. and Wilkes, D. (1991). Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*,7: 859–865.

Dudek, G., Jenkin, M., Milios, E. and Wilkes, D. (1996). A taxonomy for multi-agent robotics. *Autonomous Robots Journal*,3(4): 375–397.

Efraim, A. and D.Peleg (2007). Distributed algorithms for partitioning a swarm of autonomous mobile robots. *Structural Information and Communication Complexity* (*Lecture Notes in Computer Science*, Vol. 4474). Berlin, Springer, pp. 180–194.

Felner, A., Shoshani, Y., Altshuler, Y. and Bruckstein, A. (2006). Multi-agent physical a* with large pheromones. *Journal of Autonomous Agents and Multi-Agent Systems*, 12(1): 3–34.

Hagelbäck, J. and Johansson, S. (2008). Demonstration of multi-agent potential fields in real-time strategy games. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1687–1688.

Harmatia, I. and Skrzypczykb, K. (2009). Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework. *Robotics and Autonomous Systems*, 57(1): 75–86.

Hedberg, S. (1995). Robots cleaning up hazardous waste. *AI Expert*, pp. 20–24.

Held, M. (1991). *On the Computational Geometry of Pocket Machining* (*Lecture Notes in Computer Science*, Vol.). Berlin, Springer.

Henrich, D. (1994). Space efficient region filling in raster graphics. *The Visual Computer*, 10: 205–215.

Hollinger, G., Singh, S., Djugash, J. and Kehagias, A. (2009). Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*,28(2): 201–219.

Isler, V., Kannan, S. and Khanna, S. (2006). Randomized pursuit-evasion with local visibility. *SIAM Journal of Discrete Mathematics*,20: 26–41.

Itai, A., Papadimitriou, C. and Szwarefiter, J. (1982). Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11: 676–686.

Janakiraman, R., Waldvogel, M., and Zhang, Q. (2003). Indra: a peer-to-peer approach to network intrusion detection and prevention. *Proceedings of IEEE WETICE*.

Klos, T. and van Ahee, G. (2008). Evolutionary dynamics for designing multi-period auctions. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1589–1592.

Koenig, S., Likhachev, M. and Sun, X. (2007). Speeding up moving-target search. *AAMAS '07: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. New York, ACM Press, pp. 1–8.

Koenig, S. and Liu, Y. (2001). Terrain coverage with ant robots: A simulation study. *AGENTS'01*.

Koopman, B. (1980). *Search and Screening: General Principles with Historical Applications*. Oxford, Pergamon Press.

Manisterski, E., Lin, R. and Kraus, S. (2008). Understanding how people design trading agents over time. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1593–1596.

Mastellone, S., Stipanovi, D., Graunke, C., Intlekofer, K. and Spong, M. (2008). Formation control and collision avoidance for multi-agent non-holonomic systems: theory and experiments. *The International Journal of Robotics Research*, 27(1): 107–126.

Michael, N., Zavlanos, M., Kumar, V. and Pappas, G. (2008). Distributed multi-robot task assignment and formation control. *IEEE International Conference on Robotics and Automation, 2008 (ICRA 2008)*, pp. 128–133.

Morse, P. and Kimball, G. (1951). *Methods of Operations Research*. New York, MIT Press/Wiley.

Neubauer, W. (1993). Locomotion with articulated legs in pipes or ducts. *Robotics and Autonomous Systems*, 11: 163–169.

Pagello, E., D'Angelo, A., Montesello, F., Garelli, F. and Ferrari, C. (1999). Cooperative behaviors in multi-robot systems through implicit communication. *Robotics and Autonomous Systems*,29(1): 65–77.

Polycarpou, M., Yang, Y. and Passino, K. (2001). A cooperative search framework for distributed agents. *IEEE International Symposium on Intelligent Control*, 1–6.

Rehak, M., Pechoucek, M., Celeda, P., Krmicek, V., Grill, M. and Bartos, K. (2008). Multi-agent approach to network intrusion detection. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1695–1696.

Rekleitis, I., Lee-Shuey, V., Newz, A. P. and Choset, H. (2004). Limited communication, multi-robot team based coverage. *IEEE International Conference on Robotics and Automation*.

Russell, R. (1993). Mobile robot guidance using a short-lived heat trail. *Robotica*,11: 427–431.

Sawhney, R., Krishna, K., Srinathan, K., and Mohan, M. (2008). On reduced time fault tolerant paths for multiple UAVs covering a hostile terrain. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1171–1174.

Sen, S., Sekaran, M., and Hale, J. (1994). Learning to coordinate without sharing information. *Proceedings of AAAI*, pp. 426–431.

Shoham, Y. and Tennenholtz, M. (1995). On social laws for artificial agent societies: Off line design. *AI Journal*, 73(1–2): 231–252.

Simon, H. (1981). *The Sciences of the Artificial*, 2nd edn. Cambridge, MA, MIT Press.

Steels, L. (1990). Cooperation between distributed agents through self-organization. *Decentralized A.I.—Proceedings of the First European Workshop on Modeling Autonomous Agents in Multi-Agents World*, DeMazeau, Y. and Muller, J. P. (eds). Amsterdam, Elsevier, pp. 175–196.

Su, H., Wang, X. and Lin, Z. (2009). Flocking of multi-agents with a virtual leader. *IEEE Transactions on Automatic Control*, 54(2): 293–307.

Svennebring, J. and Koenig, S. (2004). Building terrain-covering ant robots: A feasibility study. *Autonomous Robots*,16(3): 313–332.

Thorndike, A. (1946). *Summary of Antisubmarine Warfare Operations in World War II*. Summary Report, NDRC Summary Report.

Thrun, S. B. (1992). *Efficient Exploration in Reinforcement Learning*. Technical Report CMU-CS-92-102, Carnegie Mellon University.

Trajkovski, G. and Collins, S. (2009). *Handbook of Research on Agent-Based Societies: Social and Cultural Interactions*. Idea Group Inc (IGI).

Vainsencher, D. and Bruckstein, A. M. (2008). On isoperimetrically optimal polyforms. *Theoretical Computer Science*, 406(1–2): 146–159.

Vincent, P. and Rubin, I. (2004). A framework and analysis for cooperative search using UAV swarms. *ACM Symposium on Applied Computing*.

Visser, U., Ribeiro, F., Ohashi, T. and Dellaert, F. (eds.) (2008). *RoboCup 2007: Robot Soccer World Cup XI* (*Lecture Notes in Computer Science*, Vol. 5001). Berlin, Springer.

Wagner, I., Altshuler, Y., Yanovski, V. and Bruckstein, A. (2008). Cooperative cleaners: a study in ant robotics. *The International Journal of Robotics Research*, 27(1): 127–151.

Wagner, I. and Bruckstein, A. (1994). *Row Straightening via Local Interactions*. Technical Report CIS-9406, Center for Intelligent Systems, Technion, Haifa, Israel.

Wagner, I. and Bruckstein, A. (2001). From ants to a(ge)nts: a special issue on ant robotics. *Annals of Mathematics and Artificial Intelligence*, 31(1–4): 1–6.

Wagner, I., Lindenbaum, M. and Bruckstein, A. (1998). Efficiently searching a graph by a smell-oriented vertex process. *Annals of Mathematics and Artificial Intelligence*, 24: 211–223.

Weitzenfeld, A. (2008). A prey catching and predator avoidance neural-schema architecture for single and multiple robots. *Journal of Intelligent and Robotic Systems*, 51(2): 203–233.

Work, H., Chown, E., Hermans, T. and Butterfield, J. (2008). Robust team-play in highly uncertain environments. *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1199–1202.

Yanovski, V., Wagner, I. and Bruckstein, A. (2001). Vertex-ants-walk: a robust method for efficient exploration of faulty graphs. *Annals of Mathematics and Artificial Intelligence*, 31(1–4): 99–112.

Zheng, X. and Koenig, S. (2007). Robot coverage of terrain with non-uniform traversability. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 3757–3764.

Zheng, X. and Koenig, S. (2008). Reaction functions for task allocation to cooperative agents. *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 559–566.