Static and expanding grid coverage with ant robots: Complexity results

Yaniv Altshuler [a,b,*], Alfred M. Bruckstein [a]

[a] *Technion — Israel Institute of Technology, Technion City, Haifa 32000, Israel*
[b] *Deutsche Telekom Laboratories, Ben Gurion University of the Negev, POB 653, Beer Sheva 84105, Israel*

A B S T R A C T

In this paper we study the strengths and limitations of collaborative teams of simple agents. In particular, we discuss the efficient use of "ant robots" for covering a connected region on the $\mathbf{Z}^2$ grid, whose area is unknown in advance, and which expands at a given rate, where n is the initial size of the connected region. We show that regardless of the algorithm used, and the robots' hardware and software specifications, the minimal number of robots required in order for such a coverage to be possible is $\Omega(\sqrt{n})$. In addition, we show that when the region expands at a sufficiently slow rate, a team of $\Theta(\sqrt{n})$ robots could cover it in at most $O(n^2 \ln n)$ time. This completion time can even be achieved by myopic robots, with no ability to directly communicate with each other, and where each robot is equipped with a memory of size $O(1)$ bits w.r.t. the size of the region (therefore, the robots cannot maintain maps of the terrain, nor plan complete paths). Regarding the coverage of non-expanding regions in the grid, we improve the current best known result of $O(n^2)$ by demonstrating an algorithm that guarantees such a coverage with completion time of $O(\frac{1}{k}n^{1.5} + n)$ in the worst case, and faster for shapes of perimeter length which is shorter than $O(n)$.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Motivation. In nature, ants, bees and 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.

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, e.g. [1–3].

Swarm based robotic systems can generally be defined as highly decentralized collectives, i.e., groups of extremely simple robotic 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 [4,5], patrolling [6,7], fault-tolerant cooperation [8,9], network security [10], swarm control [11,12], design of mission plans

---

* Corresponding author at: Technion −Israel Institute of Technology, Technion City, Haifa 32000, Israel. Tel.: +972 4 8294898; fax: +972 4 8293900.
E-mail addresses: yanival@cs.technion.ac.il (Y. Altshuler), freddy@cs.technion.ac.il (A.M. Bruckstein).
URLs: http://www.cs.technion.ac.il/~yanival (Y. Altshuler), http://www.cs.technion.ac.il/~freddy (A.M. Bruckstein).

[13,14], role assignment [15–17], multi-robot path planning [7,18,19], traffic control [20,21], formation generation [22–24], formation keeping [25,26], exploration and mapping [27–29], target tracking [30,31] and distributed search, intruder detection and surveillance [32,33].

Unfortunately, the mathematical/geometrical theory of such multi-agent systems is far from being satisfactory, as pointed out in [34–37] and many other papers.

Multi-robotics in dynamic environments. The vast majority of the works mentioned above discuss challenges involving a multi-agent system operating on static domains. Such models, however, are often too limited to capture "real world" problems which, in many cases, involve external element, which may influence their environment, activities and goals. Designing robotic agents that can operate in such environments presents a variety of 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 unreliable, due to the changes that take place in the environment. Task allocation, cellular decomposition, domain learning and other approaches often used by multi-agents systems all become impractical, at least to some extent. Hence, the agents' behavior must ensure that the agents generate a desired effect, regardless 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 while being searched for by the searching robots, respond by performing various evasive maneuvers intended to prevent their interception. This problem, dating back to World War II operations research (see e.g. [38,39]), requires the robotic agents to cope with a search area that expands while scanned. The first documented example for search in dynamic domains discussed a planar search problem, considering the scanning of a corridor between parallel borders. This problem was solved in [40] in order to determine optimal strategies for aircraft searching for moving ships in a channel.

A similar problem was presented in [41], 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 assumed to be adversarial and equipped with strong sensors, capable of telling the locations of the UAVs from very long distances. The search strategy suggested in [41] 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 [42], where an improved decentralized search strategy was discussed, demonstrating nearly optimal completion time, compared to the theoretical optimum achievable by any search algorithm.

Collaborative coverage of expanding domains. In this paper we shall examine a problem in which a group of ant-like robotic agents must cover an unknown region in the grid, that possibly expands over time. This problem is also strongly related to the problem of distributed search after mobile and evading target(s) [42–44] or the problems discussed under the names of "Cops and Robbers" or "Lions and Men" pursuits [45–48].

We analyze such issues using the results presented in [49–51], concerning the *Cooperative Cleaners* problem, a problem that assumes a regular grid of connected 'pixels' / 'tiles' / 'squares' / 'rooms', part 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. In the dynamic variant of this problem a deterministic evolution of the environment in assumed, simulating a spreading *contamination* (or spreading *fire*). In the spirit of [52] we consider simple robots with only a bounded amount of memory (i.e., *finite-state-machines*).

First, we discuss the collaborative coverage of static grids. We demonstrate that the best completion time known to date ($O(n^2)$, achievable for example using the *LRTA** search algorithm [53]) can be improved to guarantee grid coverage in $O(\frac{1}{k}n^{1.5} + n)$.

Later, we discuss the problem of covering an expanding domain, namely, a region in which "covered" tiles that are adjacent to "uncovered" tiles become "uncovered" every once in a while. Note that the grid is infinite, namely although initial size of the region is n, it can become much greater over time. We show that using any conceivable algorithm, and using as sophisticated and potent robotic agents as possible, the minimal number of robots below which covering such a region is impossible equals $\Omega(\sqrt{n})$. We then show that when the region expands sufficiently slowly, specifically, every $O(\frac{c_0}{\gamma_1})$ time steps (where c_0 is the circumference of the region and where γ_1 is a geometric property of the region, which ranges between $O(1)$ and $O(\ln n)$), a group of $\Theta(\sqrt{n})$ robots can successfully cover the region. Furthermore, we demonstrate that in this case a cover time of $O(n^2 \ln n)$ can be guaranteed.

These results are the first analytic results ever concerning the complexity of the number of robots required to cover an expanding grid, as well as for the time such a coverage requires.

2. Related work

In general, most of the techniques used for the task of a distributed coverage use some sort of cellular decomposition. For example, in [54] the area to be covered is divided between the agents based on their relative locations. In [55] 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 [56], 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 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.

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 [55–57]), most of them lack analytic bounds for the coverage time (although in many cases an extensive amount of empirical results of this nature are made available by extensive simulations).

An interesting work discussing a decentralized coverage of terrains is presented in [58]. 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 [59]. 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.

A search for analytic results concerning the completion time of ant robots covering an area in the grid revealed only a handful of works. The main result in this regard is that of [60,61], 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 (using integer markers that are placed on the graph's nodes), and analytic bounds for the coverage time are discussed. Based on a more general result for strongly connected undirected graphs shown in [62,63], the cover time of teams of ant robots (of a given size) that use node counting is shown to be $t_k(n) = O(n^{\sqrt{n}})$, when $t_k(n)$ is the cover time of a region of size n using k robots. It should be mentioned though, that in [60] the authors clearly state that it is their belief that the coverage time for robots using node counting in grids is much smaller. This evaluation is also demonstrated experimentally. However, no analytic evidence for this was available thus far.

Another algorithm to be mentioned in this scope is the *LRTA** search algorithm. This algorithm was first introduced in [53] and its multi-robotics variant is shown in [62] to guarantee cover time of undirected connected graphs in polynomial time. Specifically, on grids this algorithm is shown to guarantee coverage in $O(n^2)$ time (again, using integer markers).

Vertex-Ant-Walk, a variant of the node counting algorithm presented in [64], is shown to achieve a coverage time of $O(n\delta_G)$, where δ_G is the graph's diameter. Specifically, the cover time of regions in the grid is expected to be $O(n^2)$ (however for various "round" regions, a cover time of approximately $O(n^{1.5})$ can be achieved). This work is based on a previous work in which a cover time of $O(n^2\delta_G)$ was demonstrated [65].

Another work called *Exploration as Graph Construction*, which provides a coverage of degree bounded graphs in $O(n^2)$ time, can be found in [66]. Here a group of limited ant robots explore an unknown graph using special "markers".

Interestingly, a similar performance can be obtained by using the simplest algorithm for multi-robots navigation, namely, random walk. Although in general undirected graphs a group of k random walking robots may require up to $O(n^3)$ time, in degree bounded undirected graphs such robots would achieve a much faster covering, and more precisely, $O\left(\frac{|E|^2 log^3 n}{k^2}\right)$ [67].

For regular graphs or degree bounded planar graphs a coverage time of $O(n^2)$ can be achieved [68], although in such a case there is also a lower bound for the coverage time, which equals $\Omega(n(\log n)^2)$ [69].

We next show that the problem of collaborative coverage in static grid domains can be completed in $O(\frac{1}{k}n^{1.5} + n)$ time and that collaborative coverage of dynamic grid domains can be achieved in $O(n^2 \ln n)$.

3. The dynamic cooperative cleaners problem

Following is a short summary of the *Cooperative Cleaners* problem, as it appears in [49] (static variant) and [50,51] (dynamic variant).

We shall assume that the time is discrete. Let the undirected graph $G(V, E)$ denote a two-dimensional integer grid $\mathbf{Z}^2$, whose vertices (or "*tiles*") have a binary property called '*contamination*'. Let $cont_t(v)$ state the contamination state of the tile v at time t, taking either the value "*on*" or "*off*". Let F_t be the contaminated sub-graph of G at time t, i.e.: $F_t = \{v \in G \mid cont_t(v) = on\}$. We assume that F_0 is a single connected component. Our algorithm will preserve this property along its evolution.

Let a group of k robots that can move on the grid G (moving from a tile to its neighbor in one time step) be placed at time t_0 on F_0, at point $p_0 \in F_t$. Each robot is equipped with a sensor capable of telling the contamination status of all tiles in the digital sphere of diameter 7, surrounding the robot (namely, in all the tiles that their Manhattan distance from the robot is equal or smaller than 3. See an illustration in Fig. 1). A robot is also aware of other robots which are located in these tiles, and all the robots agree on a common direction. Each tile may contain any number of robots simultaneously. Each robot is equipped with a memory of size $O(1)$ bits.[1] When a robot moves to a tile v, it has the possibility of cleaning this

---

[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 [49] 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.
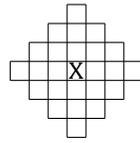
Fig. 1. An illustration of a digital sphere of diameter 7, placed around a robot.
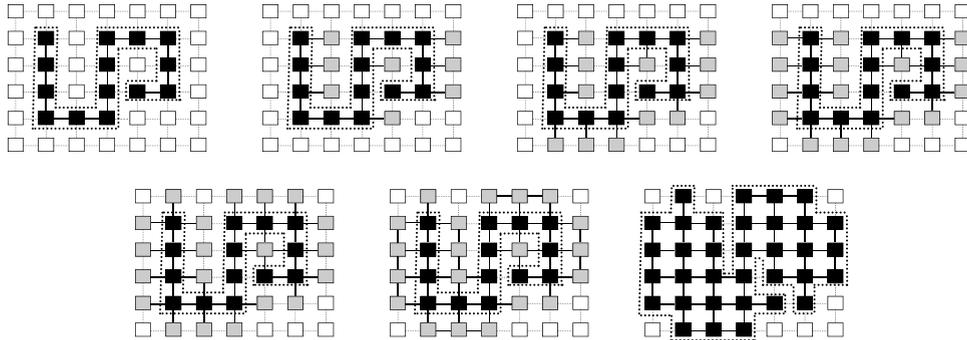

Fig. 2. A demonstration of the barrier expansion process as a result of a contamination spread.

tile (i.e., causing $cont(v)$ to become *off*. The robots 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.

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. This barrier is intended to guarantee the preservation of the simple connectivity of F_t, crucial for the operation of the robots, due to their limited memory. When a robot cleans a contaminated tile, the barrier retreats, in order to fit the void previously occupied by the cleaned tile. Every d time steps, the contamination spreads. That is, if $t = nd$ for some positive integer n, then:

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

Here, the term $4 - Neighbors(v)$ simply means the four tiles adjacent to tile v (namely, the tiles whose Manhattan distance from v equals 1). While the contamination spreads, the elastic barrier stretches while preserving the simple connectivity of the region, as demonstrated in Fig. 2. For the robots who travel along the tiles of F, the barrier signals the boundary of the contaminated region.

The robots' goal is to clean G by eliminating the contamination entirely.

It is important to note that no central control is allowed, and that the system is fully decentralized (i.e., all robots are identical and no explicit communication between the robots is allowed). An important advantage of this approach, in addition to the simplicity of the robots, is fault tolerance—even if almost all the robots die and evaporate before completion, the remaining ones will eventually complete the mission, if possible.

A survey of previous results. The cooperative cleaners problem was previously studied in [49] (static version) and [50,51] (dynamic version). A cleaning algorithm called **SWEEP** was proposed (used by a decentralized group of simple mobile robots, for exploring and cleaning an unknown "contaminated" sub-grid F, expanding every d time steps) and its performance analyzed.

The **SWEEP** algorithm is based on a constant traversal of the contaminated region, preserving the connectivity of the region while cleaning all *non-critical points*—points which when cleaned disconnect the contaminated region. Using this algorithm the agents are guaranteed to stop only upon completing their mission. The algorithm can be implemented using only local knowledge, and local interactions by immediately adjacent agents. 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 Fig. 3.

In order to formally describe the **SWEEP** algorithm, we should first define several additional terms. Let $\tau(t) = \big(\tau_1(t), \tau_2(t), \ldots, \tau_k(t)\big)$ denote the locations of the k agents at time t. In addition, 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:

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

The term ∂F denotes the boundary of F, defined via:

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

The term '*rightmost*' can now be defined as follows:
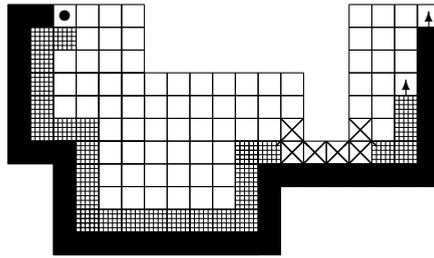
**Fig. 3.** An example of two agents using the **SWEEP** protocol, at time step 40 (with contamination spreading speed $d > 40$). All the tiles presented were contaminated at time 0. The black dot denotes the starting point of the agents. The X's mark the *critical points* which are not cleaned. The black tiles are the tiles cleaned by the first agent. The second layer of marked tiles represent the tiles cleaned by the second agent.
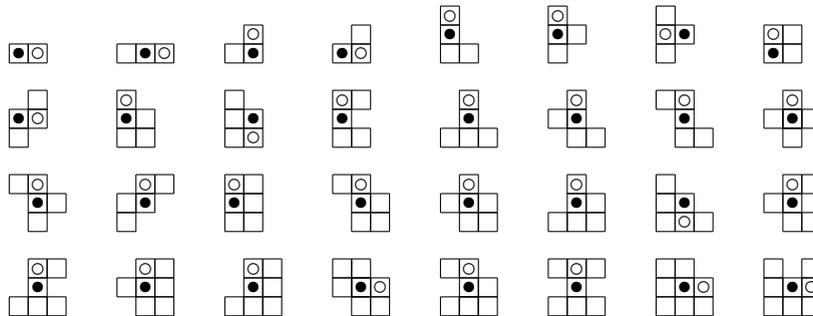
Fig. 4. When $t = 0$ the first movement of an agent located in (x, y) should be decided according to initial contamination status of the neighbors of (x, y), as it appears in these charts—the agent's initial location is marked with a filled circle while the destination is marked with an empty one. All 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.

- If $t = 0$ then select the tile as instructed in Fig. 4.
- 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 not $\tilde{\tau}_i(t) \in \partial F_t$ 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.

A schematic flowchart of the protocol, describing its major components and procedures is presented in Fig. 5. The complete pseudo-code of the protocol and its sub-routines appears in Figs. 6 and 7. Upon initialization of the system, the *System Initialization* procedure is called (defined in Fig. 6). This procedure sets various initial values of the agents, and calls the protocol's main procedure—*SWEEP* (defined in Fig. 7). This procedure in turn, uses various sub-routines and functions, all defined in Fig. 6. The *SWEEP* procedure is comprised of a loop which is executed continuously, until detecting one of the 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 the entire contaminated tiles. Until these conditions are met, each agent goes through the following sequence of commands. First each agent calculates its desired destination at the current turn. Then, each agent calculated 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 towards the same direction, the agent who had entered the tile first gets to leave the tile, while the other agents wait. In case 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 by using the *waiting dependencies* mechanism, which is implemented by each agent via an internal positioning of itself in a local ordering of his 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.

Following are several results that we will later use. While using these results, we note that completely *cleaning* a region is at least as strong as *covering* it, as the number of "uncovered" tiles at any given time can be modeled by the number of "contaminated" tiles, since the number of uncovered tiles in the original region to be explored is clearly upper bounded at all times by the number of remaining contaminated tiles that belong to this region.

Fig. 5. 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 (either as a result of *waiting* or *resting* dependencies—see lines 6 and 14 of **SWEEP** for more details).

```
 1: System Initialization
 2:     Arbitrarily choose a pivot point p₀ in ∂F₀, and mark it as critical point
 3:     Place all the agents on p₀
 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 (x₀, y₀) to (x₁, y₁) */
19:     priority ← 2(x₁ − x₀) + (y₁ − y₀)
20: End procedure


21: Check Completion of Mission
22:     If ((x, y) = p₀) 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 Fₜ 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 ∂Fₜ contains at least 2 agents */
35:     saturated perimeter ← false
36:     If ((x, y) ∈ ∂Fₜ) and both (x, y) and all of its non-critical neighbors
        in ∂Fₜ contain at least two agents then
37:         saturated perimeter ← true
38:     If ((x, y) ∈ ∂Fₜ) and both (x, y) and all of its neighbors in ∂Fₜ has
        saturated perimeter = true then
39:         Ignore resting commands for this time step
40: End procedure
```

Fig. 6. The first part of the **SWEEP** cleaning protocol.

1: **SWEEP Protocol** /* Controls agent i after **Agent Reset** */
2:     **Check Completion of Mission**
3:     **Check "Near Completion" of Mission**
4:     *dest* ← *rightmost neighbor* of (x,y) /* Calculate destination */
5:     *destination signal bits* ← *dest* /* Signaling the desired destination */
6:     /* Calculate resting dependencies (solves agents' clustering problem) */
7:     From all agents in (x,y) except agent i we define be 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 this of agent i
10:    *resting* ← *false*
11:    **If** $(K_1 \neq \emptyset)$ or $(K_2 \neq \emptyset)$ then
12:      *resting* ← *true*
13:      **If** (current time-step $\mathcal{T}$ did not end yet) then jump to 3 **Else** jump to 3
14:    *waiting* ← $\emptyset$ /* Waiting dependencies (agents synchronization) */
15:    Let *active agent* denote a *non-resting* agent which didn't move in $\mathcal{T}$ yet
16:    **If** $(x - 1, y) \in F_t$ contains an active agent then *waiting* ← *waiting* ∪ {*left*}
17:    **If** $(x, y - 1) \in F_t$ contains an active agent then *waiting* ← *waiting* ∪ {*down*}
18:    **If** $(x - 1, y - 1) \in F_t$ contains an active agent then *waiting* ← *waiting* ∪ {$l - d$}
19:    **If** $(x + 1, y - 1) \in F_t$ contains an active agent then *waiting* ← *waiting* ∪ {$r - d$}
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* ← *waiting* ∪ {*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* ← *waiting* ∪ {*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 3 **Else** jump to 3
24:    /* Decide whether or not (x, y) should be cleaned */
25:    **If** ¬ $((x, y) \in \partial F_t)$ or $((x, y) \equiv p_0)$ or (x, y) has 2 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 does not still contain other agents
29:    Move to *dest*
30:    Wait until $\mathcal{T}$ ends.
31:    Return to 3

Fig. 7. The **SWEEP** cleaning protocol.

Result 1 (*Cleaning a Non-Expanding Contamination*)**.** *The time it takes for a group of K robots using the **SWEEP** algorithm to clean a region F of the grid is at most:*

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

Here $W(F)$ denotes the depth of the region F (the shortest path from some internal point in F to its boundary, for the internal point whose shortest path is the longest) and as defined above, ∂F denotes the boundary of F, defined via:

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

The term $8 - Neighbors(v)$ is used to denote the eight tiles that tile v is immediately surrounded by.

Result 2 (*Universal Lower Bound on Contaminated Area*)**.** *Using any cleaning algorithm, the area at time t of a contaminated region that expands every d time steps can be recursively lower bounded, as follows:*

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

Here S_t denotes the area of the contaminated region at time t (such that $S_0 = n$).

Result 3 (*Upper Bound on Cleaning Time for SWEEP on Expanding Domains*)**.** *For a group of k robot using the **SWEEP** algorithm to clean a region F on the grid, that expands every d time steps, the time it takes the robots to clean F is at most d multiplied by the minimal positive value of the following two numbers:*

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

where:

$$A_1 = \frac{c_0 + 2 - \gamma_2}{4}, \qquad A_2 = \frac{c_0 + 2 + \gamma_2}{4}, \qquad A_3 = \frac{8 \cdot \gamma_2}{d \cdot k},$$

$$A_4 = \gamma_1 - \frac{\gamma_2 \cdot \gamma}{d}, \qquad \gamma_1 = \psi\,(1 + A_2) - \psi\,(1 + A_1)$$

$$\gamma_2 = \sqrt{(c_0 + 2)^2 - 8S_0 + 8}$$

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

Here c_0 is the circumference of the initial region F_0, and where $\psi(x)$ is the *Digamma* function (studied in [70])—the logarithmic derivative of the *Gamma function*, defined as:

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

or as:

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

Note that although $c_0 = O(|\partial F|)$ the actual length of the perimeter of the region can be greater than its cardinality, as several tiles may be traversed more than once. In fact, in [49] it is shown that $c_0 \leq 2 \cdot |\partial F| - 2$.

4. Grid coverage—analysis

We note again that when discussing the coverage of regions on the grid, either static or expanding, it is enough to show that the region can be cleaned by the team of robots, as clearly the cleaned sites are always a subset of the visited ones. We first present the cover time of a group of robots operating in non-expanding domains, using the **SWEEP** algorithm.

Theorem 1. *Given a connected region of $S_0 = n$ grid tiles and perimeter c_0, that should be covered by a team of k ant-like robots, the robots can cover it using $O\left(\frac{1}{k}S_0^{1.5} + S_0\right)$ time.*

Proof. Since $|\partial F_0| = \Theta(c_0)$, and $W(F_0) = O(\sqrt{S_0})$, recalling Result 1 we can see that:

$$t_k(n) = t_{static}(k) = O\left(\frac{1}{k}\sqrt{S_0} \cdot c_0 + c_0 + k \right).$$

As $c_0 = O(S_0)$ and for practical reasons we assume that $k < n$ this would equal in the worse case to:

$$t_k(n) = t_{static}(k) = O\left(\frac{1}{k}S_0^{1.5} + S_0 \right). \quad \square$$

We now examine the problem of covering expanding domains. The lower bound for the number of robots required for completing is as follows.

Theorem 2. *Given a region of size $S_0 \geq 3$ tiles, expanding every d time steps, a team of less than $\frac{\sqrt{S_0}}{d}$ robots cannot clean the region, regardless of the algorithm used.*

Proof. Recalling Result 2 we can see that:

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

By assigning $k = \frac{\sqrt{S_0}}{d}$ we can see that:

$$\Delta S_t = S_{t+d} - S_t \geq \left\lfloor 2\sqrt{2 \cdot (S_t - \sqrt{S_0}) - 1} \right\rfloor - \sqrt{S_0}.$$

For any $S_0 \geq 3$, we see that $\Delta S_0 > 0$. In addition, for every $S_0 \geq 3$ we can see that $\frac{dS_t}{dt} > 0$ for every $t \geq 0$. Therefore, for every $S_0 \geq 3$ the size of the region will be forever growing. $\square$

Corollary 1. *Given a region of size S_0 tiles, expanding every d time steps, where $d = O(1)$ w.r.t S_0, then a team of less than $\Omega(\sqrt{S_0})$ robots cannot clean the region, regardless of the algorithm used.*

Theorem 3. *Given a region F of size S_0 tiles, expanding every d time steps, where $R(F)$ is the perimeter of the bounding rectangle of the region F, a team of k robots that at $t = 0$ are located at the same tile cannot clean the region, regardless of the algorithm used, as long as $d^2 k < \Omega(R(F))$.*

Proof. For every $v \in F$ let $l(v)$ denote the maximal distance between v and any of the tiles of F, namely:

$$l(v) = \max\{d(v, u) | u \in F\}.$$

Let $C(F) = l(v_c)$ such that $v_c \in F$ is the tile with minimal value of $l(v)$.

Let v_s denote the tile the agents are located in at $t = 0$. Let $v_d \in F$ denote some contaminated tile such that $d(v_s, v_d) = l(v_s)$. Regardless of the algorithm used by the agents, until some agent reaches v_d there will pass at least $l(v_s)$ time steps. Let us assume w.l.o.g that v_d is located to the right (or of the same horizontal coordinate) and to the top (or of the same vertical coordinate) of v_s. Then by the time some agent is able to reach v_d there exists an upper-right quarter of a digital sphere of radius $\left\lfloor \frac{l(v_s)}{d} \right\rfloor + 1$, whose center is v_d.

The number of tiles in such a quarter of digital sphere equals:

$$\frac{1}{2} \left\lfloor \frac{l(v_s)}{d} \right\rfloor^2 + \frac{3}{2} \left\lfloor \frac{l(v_s)}{d} \right\rfloor + 1 = \Theta \left(\frac{l(v_s)^2}{d^2} \right).$$

It is obvious that the region cannot be cleaned until v_d is cleaned. Let t_d denote the time at which the first agent reaches v_d. It is easy to see that $t_d \geq l(v_s)$. Therefore, regardless of activities of the agents until time step t_d, there are now k agents that has to clean a region of at least $\Theta \left(\frac{l(v_s)^2}{d^2} \right)$ tiles, spreading every d time steps. Using Theorem 2 we know that k agents cannot clean an expanding region of $k = \frac{\sqrt{S_0}}{d}$ tiles. Namely, at time t_d k agents could not clean the contaminated tiles if:

$$d^2 k < \Omega \left(l(v_s) \right).$$

As $l(v_s) \geq C(F)$ we know that k agents could not clean an expanding contaminated region where: $d^2 k < \Omega \left(C(F) \right)$. It is easy to see that for every region F, if $R(F)$ is the length of the perimeter of the bounding rectangle of F then $C(F) = \Theta(R(F))$. □

Lemma 1. *For every connected region of size $S_0 \geq 3$ and perimeter of length c_0:*

$$\frac{1}{2} c_0 < \gamma_2 < c_0.$$

Proof. Let us assume by contradiction that $(c_0 + 2)^2 \leq (8S_0 + 8)$. This means $c_0 \leq \sqrt{8S_0 + 8} - 2$. However, the minimal circumference of a region of size S_0 is achieved when the region is arranged in the form of an 8-connected digital sphere, in which case $c_0 \geq 4\sqrt{S_0} - 4$, contradicting the assumption that $c_0 \leq \sqrt{8S_0 + 8} - 2$ for every $S_0 > 5$. Therefore, $\gamma_2 \in \mathbb{R}$.

Let us assume by contradiction that $\gamma_2 < \frac{1}{2} c_0$. Therefore:

$$(c_0 + 2)^2 - 8S_0 + 8 < \frac{1}{4} c_0^2$$

which implies:

$$c_0 < -\frac{16}{6} + \sqrt{10\frac{2}{3} S_0 - 8\frac{8}{9}} < 3.266\sqrt{S_0} - 2.$$

However, we know that $c_0 \geq 4\sqrt{S_0} - 4$, which contradicts the assumption that $\gamma_2 < \frac{1}{2} c_0$ for every $S_0 \geq 3$.

Let us assume by contradiction that $\gamma_2 > c_0$. Therefore:

$$(c_0 + 2)^2 - 8S_0 + 8 > c_0^2$$

which implies:

$$c_0 > 4S_0 - 6.$$

However, we know that $c_0 \leq 2S_0 - 2$ (as c_0 is maximized when the tiles are arranged in the form of a straight line), contradicting the assumption that $\gamma_2 > c_0$ for every $S_0 \geq 3$. □

Lemma 2. *For every connected region of size $S_0 \geq 3$ and perimeter of length c_0:*

$$\Omega(1) < \gamma_1 < O(\ln n).$$

Proof. Let us observe γ_1:

$$\gamma_1 \triangleq \psi \left(1 + \frac{c_0 + 2 + \gamma_2}{4} \right) - \psi \left(1 + \frac{c_0 + 2 - \gamma_2}{4} \right).$$

From Lemma 1 we can see that $1 < \left(1 + \frac{c_0 + 2 - \gamma_2}{4} \right) < \frac{1}{4} c_0$. Note that $\psi(1) = -\hat{\gamma}$ where $\hat{\gamma}$ is the *Euler–Mascheroni* constant, defined as:

$$\hat{\gamma} = \lim_{n \to \infty} \left[\left(\sum_{k=1}^{n} \frac{1}{k} \right) - \log(n) \right] = \int_{1}^{\infty} \left(\frac{1}{\lfloor x \rfloor} - \frac{1}{x} \right) dx$$

which equals approximately 0.57721. In addition, $\psi(x)$ is monotonically increasing for every $x > 0$. As we also know that $\psi(x)$ is upper bounded by $O(\ln x)$ for large values of x, we see that:

$$-0.58 < \psi\left(1 + \frac{c_0 + 2 - \gamma_2}{4}\right) < O(\ln n). \tag{4.1}$$

From Lemma 1 we also see that $1 < \left(1 + \frac{c_0 + 2 + \gamma_2}{4}\right) < \frac{1.5}{4}c_0$ meaning that:

$$\psi\left(1 + \frac{c_0 + 2 + \gamma_2}{4}\right) = \Theta(\ln n). \tag{4.2}$$

Combining Eqs. (4.1) and (4.2) we see that:

$$\Omega(1) < \gamma_1 < O(\ln n). \quad \square \tag{4.3}$$

Theorem 4. *Result 3 returns a positive real number for the covering time of a region of S_0 tiles that expands every d time steps, when the number of robots is $\Theta(\sqrt{S_0})$ and $d = \Omega(\frac{c_0}{\gamma_1})$, and where γ_1 shifts from $O(1)$ to $O(\ln S_0)$ as c_0 grows from $O(\sqrt{S_0})$ to $O(S_0)$, defined as:*

$$\gamma_1 = \psi\left(1 + \frac{c_0 + 2 + \gamma_2}{4}\right) - \psi\left(1 + \frac{c_0 + 2 - \gamma_2}{4}\right)$$
$$\gamma_2 = \sqrt{(c_0 + 2)^2 - 8S_0 + 8}.$$

Proof. Following are the requirements that must hold in order for Result 3 to yield a real number:

- $d \cdot k \neq 0$
- $|\partial F| > 1$
- $A_3 \neq 0$
- $(c_0 + 2)^2 > 8S_0 - 8$
- $(A_1A_3 - A_4)^2 \geq 4A_3(A_2 - A_1 - A_1A_4)$.

The first and second requirements hold for every non-trivial scenario. The third requirement is implied by the fourth. The fourth assumption is a direct result of Lemma 1.

As for the last requirement, we ask whether:

$$A_1^2 A_3^2 + A_4^2 \geq 4A_2A_3 - 4A_1A_3 - 2A_1A_3A_4$$

which subsequently means that we must have:

$$\begin{array}{l}\frac{\gamma_2^2}{d^2k^2}\left(c_0^2 + \gamma_2^2 - c_0\gamma_2\right) \\ +\gamma_1^2 + \frac{\gamma_2^2\gamma^2}{d} - \frac{\gamma\gamma_1\gamma_2}{d}\end{array} \geq 4\frac{\gamma_2}{dk}\left(\begin{array}{c}4\gamma_2 - c_0\gamma_1 + \\ c_0\frac{\gamma_2\gamma}{d} - 2\gamma_1 + 2\frac{\gamma_2\gamma}{d} + \\ \gamma_1\gamma_2 - \frac{\gamma_2^2\gamma}{d}\end{array}\right).$$

Using Lemmas 1 and 2 we should make sure that:

$$\frac{\gamma_2^2}{dk^2}c_0^2 + \gamma_1^2 d + \gamma_2^2\gamma^2 - \gamma\gamma_1\gamma_2 \geq O\left(\frac{c_0\gamma_2\gamma_1}{k} + \frac{c_0\gamma_2^2\gamma}{dk}\right).$$

Using $W(F) = O(\sqrt{S_0})$ and $\Omega(\sqrt{S_0}) = |\partial F| = O(S_0)$ and dividing by γ_2^2 (which we know to be larger than zero), we can write the above as follows:

$$\frac{c_0^2}{dk^2} + \frac{k^2 + d\ln^2 S_0}{c_0^2} + 1 \geq O\left(\frac{\ln S_0}{c_0} + \frac{k\ln S_0}{c_0^2} + \frac{\ln S_0}{k} + \frac{c_0\sqrt{S_0}}{dk^2} + \frac{c_0}{dk} + \frac{1}{d}\right).$$

As $c_0 \geq \sqrt{S_0}$ then $\frac{c_0^2}{dk^2} \geq \frac{c_0\sqrt{S_0}}{dk^2}$. In addition, $1 \geq \frac{1}{d}$ and also $1 \geq \frac{\ln S_0}{c_0}$ and $1 \geq \frac{\ln S_0}{k}$ (as Eq. (4.6) shows that $k \geq \ln S_0$). In order to have also $1 \geq \frac{c_0}{dk}$ we must have:

$$d \cdot k = \Omega(c_0). \tag{4.4}$$

In addition, we should also require that the result μ would be positive (as it denotes the coverage time). Namely, that:

$$A_4 + \sqrt{(A_1A_3 - A_4)^2 - 4A_3(A_2 - A_1 - A_1A_4)} > A_1A_3.$$

For this to hold we shall merely require that:

$$A_2 - A_1 - A_1A_4 \leq 0$$

(as A_3 is known to be positive). Assigning the values of A_1, A_2, A_4, this translates to:

$$c_0 + c_0^2 \frac{\gamma}{d} \le O(c_0 \gamma_1).$$

Dividing by c_0 we can write:

$$1 + c_0 \frac{1 + \frac{\sqrt{S_0}}{k} - \frac{d}{c_0} + \frac{k}{c_0}}{d} \le O(\gamma_1).$$

Namely:

$$c_0 + \frac{c_0 \sqrt{S_0}}{k} + k \le dO(\gamma_1).$$

As c_0 is the dominant element on the left side of the inequation, we see that:

$$d = \Omega\left(\frac{c_0}{\gamma_1}\right). \tag{4.5}$$

Assigning this lower bound for d we can now see that:

$$\Omega(\sqrt{S_0}) \le k \le O(c_0). \tag{4.6}$$

Therefore, we shall select the value of k such that:

$$k = \Theta(\sqrt{S_0}).$$

This also satisfies Eq. (4.4). $\quad\square$

Theorem 5. *The time it takes a group of $k = \Theta(\sqrt{S_0})$ robots using the **SWEEP** algorithm to cover a connected region of size S_0 tiles, that expands every $d = \Omega(\frac{c_0}{\gamma_1})$ time steps, is upper bounded as follows:*

$$t_{SUCCESS} = O\left(S_0^2 \ln S_0\right)$$

where γ_1 shifts from $O(1)$ to $O(\ln S_0)$ as c_0 grows from $O(\sqrt{S_0})$ to $O(S_0)$, defined as:

$$\gamma_1 = \psi\left(1 + \frac{c_0 + 2 + \gamma_2}{4}\right) - \psi\left(1 + \frac{c_0 + 2 - \gamma_2}{4}\right)$$
$$\gamma_2 = \sqrt{(c_0 + 2)^2 - 8S_0 + 8}.$$

Proof. Recalling Result 1 we know that if:

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

then the robots could clean the region before it expands even once. In this case, the cleaning time would be $O(\frac{1}{k}\sqrt{S_0} \cdot c_0 + c_0)$ as was shown in Theorem 1. Therefore, we shall assume that:

$$\frac{8(|\partial F_0| - 1) \cdot (W(F_0) + k)}{k} + 2k \ge d. \tag{4.7}$$

Observing Result 3 we see that:

$$
\begin{aligned}
t_{SUCCESS} &= d \cdot O\left(A_1 + \frac{|A_4|}{A_3} + \sqrt{A_1^2 + \frac{|A_1 A_4| + A_1 + A_2}{A_3} + \frac{A_4^2}{A_3^2}}\right) \\
&\le d \cdot O\left(A_1 + \frac{|A_4|}{A_3} + \frac{\sqrt{A_3}\sqrt{|A_1 A_4| + A_1 + A_2}}{A_3} + \frac{|A_4|}{A_3}\right) \\
&\le d \cdot O\left(A_1 + \frac{|A_4|}{A_3} + \frac{\sqrt{|A_1 A_4|} + \sqrt{A_1} + \sqrt{A_2}}{\sqrt{A_3}}\right) \\
&= d \cdot O\left(\begin{matrix} c_0 + \gamma_2 + dk\frac{\gamma_1}{\gamma_2} + k\gamma + \\ \sqrt{k}\frac{\sqrt{c_0 + \gamma_2}\sqrt{d\gamma_1 + \gamma_2 \cdot \gamma}}{\sqrt{\gamma_2}} + \sqrt{\frac{kd}{\gamma_2}}\sqrt{c_0 + \gamma_2} \end{matrix}\right).
\end{aligned}
$$

Using the fact that $\gamma_2 = \Theta(c_0)$ (Lemma 1) we can rewrite this expression as:

$$d \cdot O\left(c_0 + dk\frac{\gamma_1}{c_0} + k\gamma + \sqrt{k}\sqrt{d\gamma_1 + c_0\gamma} + \sqrt{kd}\right). \tag{4.8}$$

Recalling Eq. (4.7), and as $W(F_0) = O(\sqrt{S_0})$, we can see that:

$$d = O\left(\frac{\sqrt{S_0} \cdot c_0}{k} + c_0 + k\right).$$

Therefore, $|\gamma|$ can now be written as:

$$|\gamma| = O\left(\frac{\sqrt{S_0}}{k} + \sqrt{S_0} + \frac{k}{\sqrt{S_0}} + 1\right).$$

In addition, remembering that $O(\sqrt{S_0}) \le c_0 \le O(S_0)$ we can rewrite the expression of Eq. (4.8) as follows:

$$d \cdot O\left(\frac{c_0 + dk\frac{\gamma_1}{c_0} + k\sqrt{S_0} + \frac{k^2}{\sqrt{S_0}} +}{\sqrt{kc_0}\sqrt{\frac{d}{c_0}\gamma_1 + \frac{\sqrt{S_0}}{k} + \sqrt{S_0} + \frac{k}{\sqrt{S_0}}} + \sqrt{kd}}\right) = d \cdot O\left(\frac{c_0 + dk\frac{\gamma_1}{c_0} + k\sqrt{S_0} + \frac{k^2}{\sqrt{S_0}} +}{\sqrt{kc_0}\left(\sqrt{\gamma_1} + \sqrt[4]{S_0}\sqrt{\frac{\gamma_1}{k}} + \sqrt[4]{S_0} + \frac{\sqrt{k}}{\sqrt[4]{S_0}}\right) + \sqrt{kd}}\right).$$

Using Lemma 2 we see that:

$$d \cdot O\left(\frac{c_0 + dk\frac{\ln S_0}{c_0} + k\sqrt{S_0} + \frac{k^2}{\sqrt{S_0}} +}{\sqrt{c_0 \ln S_0 \sqrt{S_0}} + \sqrt{c_0 k \sqrt{S_0}} + k\sqrt[4]{S_0} + \sqrt{kd}}\right) = d \cdot O\left(\frac{c_0 + k\sqrt{S_0}\ln S_0 + \frac{k^2}{\sqrt{S_0}} +}{\sqrt{c_0 \ln S_0 \sqrt{S_0}} + \sqrt{c_0 k \sqrt{S_0}}}\right). \tag{4.9}$$

Assuming that $k > O(\ln S_0)$ and as $c_0 = O(S_0)$ we can now write:

$$O\left(\begin{array}{c}\frac{S_0^{2.5}}{k} + S_0^2 \ln S_0 + S_0^{1.5}k\ln S_0 + k^2\sqrt{S_0}\ln S_0 +\\ \frac{k^3}{\sqrt{S_0}} + \frac{S_0^{2.25}}{\sqrt{k}} + S_0^{1.75}\sqrt{k} + S_0^{0.75}k^{1.5}\end{array}\right). \tag{4.10}$$

Using Eq. (4.6) we can see that this translates to:

$$O\left(S_0^{1.5}k\ln S_0\right). \quad \square \tag{4.11}$$

5. Conclusions

In this paper we have discussed the problem of cleaning or covering a connected region on the grid using a collaborate team of simple, finite-state-automata robotic agents. We have shown that when the regions are static, this can be done in $O(\frac{1}{k}\sqrt{n} \cdot c_0 + c_0 + k)$ time which equals $O(\frac{1}{k}n^{1.5} + n)$ time in the worst case, thus improving the previous results for this problem. In addition, we have shown that when the region is expanding in a constant rate (which is "slow enough"), a team of $\Theta(\sqrt{n})$ robots can still be guaranteed to clean or cover it, in $O(n^2 \ln n)$ time.

In addition, we have shown that teams of less than $\Omega(\sqrt{n})$ robots can *never* cover a region that expands every $O(1)$ time steps, regardless of their sensing capabilities, communications and memory resources employed, or the algorithm used. As to regions that expand slower than every $O(1)$ time steps, we have shown the following impossibility results. First, a region of n tiles that expands every d time steps cannot be covered by a group of k agents if $dk \le O(\sqrt{n})$. Using Theorem 4 we can guarantee a coverage when $dk = \Omega(\frac{n^{1.5}}{\ln n})$, or even for $dk = \Omega(n))$ (when the region's perimeter c_0 equals $O(n)$).

Second, a spreading region cannot be covered when d^2k is smaller than the order of the perimeter of the bounding rectangle of the region (which equals $O(n)$ in the worse case and $O(\sqrt{n})$ for shapes with small perimeters). Using Theorem 4 we can guarantee a coverage when $d^2k = \Omega(\frac{n^{2.5}}{\ln^2 n})$, or for $d^2k = \Omega(n^{1.5})$ (when the region's perimeter c_0 equals $O(n)$).

We believe that these results can be easily applied to various other problems in which a team of agents are required to operate in an expanding grid domain. For example, this result can show that a team of $\Theta(\sqrt{n})$ cops can always catch a robber (or for that matter, a group of robbers), moving (slowly) in an unbounded grid. Alternatively, robbers can be guaranteed to escape a team of less than $O(\sqrt{n})$ cops, if the area they are located in is unbounded.

References

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

[2] S. DeLoach, M. Kumar, Intelligence Integration in Distributed Knowledge Management, Idea Group Inc. (IGI), 2008, Ch. Multi-agent systems engineering: an overview and case study, pp. 207–224.

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

[4] U.V., et al., RoboCup 2007: Robot Soccer World Cup XI, in: Lecture Notes in Computer Science, vol. 5001, Springer, Berlin, 2008.

[5] R. Alami, S. Fleury, M. Herrb, F. Ingrand, F. Robert, Multi-robot cooperation in the martha project, IEEE Robotics and Automation Magazine 5 (1) (1998) 36–47.

[6] N. Agmon, S. Kraus, G. Kaminka, Multi-robot perimeter patrol in adversarial settings, in: IEEE International Conference on Robotics and Automation, ICRA 2008, 2008, pp. 2339–2345.

[7] N. Agmon, V. Sadov, G. Kaminka, S. Kraus, The impact of adversarial knowledge on adversarial planning in perimeter patrol, in: AAMAS'08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 55–62.
[8] S. Kraus, O. Shehory, G. Taase, Coalition formation with uncertain heterogeneous information, in: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, 2003, pp. 1–8.
[9] H. Work, E. Chown, T. Hermans, J. Butterfield, Robust team-play in highly uncertain environments, in: AAMAS'08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 1199–1202.
[10] M. Rehak, M. Pechoucek, P. Celeda, V. Krmicek, M. Grill, K. Bartos, Multi-agent approach to network intrusion detection, in: AAMAS'08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 1695–1696.
[11] R. Connaughton, P. Schermerhorn, M. Scheutz, Physical parameter optimization in swarms of ultra-low complexity agents, in: AAMAS'08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 1631–1634.
[12] M. Mataric, Interaction and intelligent behavior, Ph.D. Thesis, Massachusetts Institute of Technology, 1994.
[13] E. Manisterski, R. Lin, S. Kraus, Understanding how people design trading agents over time, in: AAMAS'08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 1593–1596.
[14] D. MacKenzie, R. Arkin, J. Cameron, Multiagent mission specification and execution, Autonomous Robots 4 (1) (1997) 29–52.
[15] G. Chalkiadakis, C. Boutilier, Sequential decision making in repeated coalition formation under uncertainty, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, 2008, pp. 347–354.
[16] X. Zheng, S. Koenig, Reaction functions for task allocation to cooperative agents, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, 2008, pp. 559–566.
[17] C. Candea, H. Hu, L. Iocchi, D. Nardi, M. Piaggio, Coordinating in multi-agent robocup teams, Robotics and Autonomous Systems 36 (2–3) (2001) 67–86.
[18] R. Sawhney, K. Krishna, K. Srinathan, M. Mohan, On reduced time fault tolerant paths for multiple uavs covering a hostile terrain, in: AAMAS'08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 1171–1174.
[19] A. Yamashita, M. Fukuchi, J. Ota, T. Arai, H. Asama, Motion planning for cooperative transportation of a large object by multiple mobile robots in a 3d environment, in: Proceedings of IEEE International Conference on Robotics and Automation, 2000, pp. 3144–3151.
[20] A. Agogino, K. Tumer, Regulating air traffic flow with coupled agents, in: AAMAS'08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 535–542.
[21] S. Premvuti, S. Yuta, Consideration on the cooperation of multiple autonomous mobile robots, in: Proceedings of the IEEE International Workshop of Intelligent Robots and Systems, 1990, pp. 59–63.
[22] R. Bhatt, C. Tang, V. Krovi, Formation optimization for a fleet of wheeled mobile robots a geometric approach, Robotics and Autonomous Systems 57 (1) (2009) 102–120.
[23] J. Fredslund, M. Mataric, Robot formations using only local sensing and control, in: Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation, IEEE CIRA 2001, 2001, pp. 308–313.
[24] N. Gordon, I. Wagner, A. Bruckstein, Discrete bee dance algorithms for pattern formation on a grid, in: IEEE International Conference on Intelligent Agent Technology, IAT03, 2003, pp. 545–549.
[25] K. Bendjilali, F. Belkhouche, B. Belkhouche, Robot formation modelling and control based on the relative kinematics equations, The International Journal of Robotics and Automation 24 (1) (2009) 79–88.
[26] T. Balch, R. Arkin, Behavior-based formation control for multi-robot teams, IEEE Transactions on Robotics and Automation 14 (6) (1998) 926–939.
[27] S. Sariel, T. Balch, Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments, in: Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling, 2005, pp. 27–33.
[28] M. Pfingsthorn, B. Slamet, A. Visser, RoboCup 2007: Robot Soccer World Cup XI, in: Lecture Notes in Computer Science, vol. 5001, Springer, Berlin, 2008, pp. 457–464. (Ch. A Scalable Hybrid Multi-robot SLAM Method for Highly Detailed Maps).
[29] I. Rekleitis, G. Dudek, E. Milios, Experiments in free-space triangulation using cooperative localization, in: IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, IROS, 2003.
[30] I. Harmatia, K. Skrzypczykb, Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework, Robotics and Autonomous Systems 57 (1) (2009) 75–86.
[31] L. Parker, C. Touzet, Multi-robot learning in a cooperative observation task, Distributed Autonomous Robotic Systems 4 (2000) 391–401.
[32] G. Hollinger, S. Singh, J. Djugash, A. Kehagias, Efficient multi-robot search for a moving target, The International Journal of Robotics Research 28 (2) (2009) 201–219.
[33] S. LaValle, D. Lin, L. Guibas, J. Latombe, R. Motwani, Finding an unpredictable target in a workspace with obstacles, in: Proceedings of the 1997 IEEE International Conference on Robotics and Automation, ICRA-97, 1997, pp. 737–742.
[34] A. Efraim, D. Peleg, Distributed algorithms for partitioning a swarm of autonomous mobile robots, in: Structural Information and Communication Complexity, in: Lecture Notes in Computer Science, vol. 4474, 2007, pp. 180–194.
[35] R. Olfati-Saber, Flocking for multi-agent dynamic systems: algorithms and theory, IEEE Transactions on Automatic Control 51 (3) (2006) 401–420.
[36] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, US, 1999.
[37] G. Beni, J. Wang, Theoretical problems for the realization of distributed robotic systems, in: IEEE Internal Conference on Robotics and Automation, 1991, pp. 1914–1920.
[38] A. Thorndike, Summary of antisubmarine warfare operations in world war II, Summary report, NDRC Summary Report, 1946.
[39] P. Morse, G. Kimball, Methods of Operations Research, MIT Press, Wiley, New York, 1951.
[40] B. Koopman, Search and Screening: General Principles with Historical Applications, Pergamon Press, 1980.
[41] P. Vincent, I. Rubin, A framework and analysis for cooperative search using uav swarms, in: ACM Simposium on Applied Computing, 2004.
[42] Y. Altshuler, V. Yanovsky, A. Bruckstein, I. Wagner, Efficient cooperative search of smart targets using UAV swarms, Robotica 26 (2008) 551–557.
[43] S. Koenig, M. Likhachev, X. Sun, Speeding up moving-target search, in: AAMAS '07: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, New York, NY, USA, 2007, pp. 1–8. doi:http://doi.acm.org/10.1145/1329125.1329353.
[44] R. Borie, C. Tovey, S. Koenig, Algorithms and complexity results for pursuit-evasion problems, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 2009, pp. 59–66.
[45] R. Isaacs, Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization, John Wiley and Sons, 1965, Reprinted by Dover Publications 1999.
[46] V. Isler, S. Kannan, S. Khanna, Randomized pursuit-evasion with local visibility, SIAM Journal of Discrete Mathematics 20 (2006) 26–41.
[47] A. Goldstein, E. Reingold, The complexity of pursuit on a graph, Theoretical Computer Science 143 (1995) 93–112.
[48] J. Flynn, Lion and man: the general case, SIAM Journal of Control 12 (1974) 581–597.
[49] I. Wagner, Y. Altshuler, V. Yanovski, A. Bruckstein, Cooperative cleaners: a study in ant robotics, The International Journal of Robotics Research (IJRR) 27 (1) (2008) 127–151.
[50] Y. Altshuler, A. Bruckstein, I. Wagner, Swarm robotics for a dynamic cleaning problem, in: IEEE Swarm Intelligence Symposium, 2005, pp. 209–216.

[51] Y. Altshuler, I. Wagner, A. Bruckstein, Swarm ant robotics for a dynamic cleaning problem—upper bounds, in: The 4th International Conference on Autonomous Robots and Agents, ICARA-2009, 2009, pp. 227–232.
[52] V. Braitenberg, Vehicles, MIT Press, 1984.
[53] R. Korf, Real-time heuristic search, Artificial Intelligence 42 (1990) 189–211.
[54] I. Rekleitis, V. Lee-Shuey, A. P. Newz, H. Choset, Limited communication, multi-robot team based coverage, in: IEEE International Conference on Robotics and Automation, 2004.
[55] Z. Butler, A. Rizzi, R. Hollis, Distributed coverage of rectilinear environments, in: Proceedings of the Workshop on the Algorithmic Foundations of Robotics, 2001.
[56] E. Acar, Y. Zhang, H. Choset, M. Schervish, A. Costa, R. Melamud, D. Lean, A. Gravelin, Path planning for robotic demining and development of a test platform, in: International Conference on Field and Service Robotics, 2001, pp. 161–168.
[57] M. Batalin, G. Sukhatme, Spreading out: A local approach to multi-robot coverage, in: 6th International Symposium on Distributed Autonomous Robotics Systems, 2002.
[58] X. Zheng, S. Koenig, Robot coverage of terrain with non-uniform traversability, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, IROS, 2007, pp. 3757–3764.
[59] X. Zheng, S. Jain, S. Koenig, D. Kempe, Multi-robot forest coverage, in: In Proc. IROS, press, 2005, pp. 3852–3857.
[60] J. Svennebring, S. Koenig, Building terrain-covering ant robots: a feasibility study, Autonomous Robots 16 (3) (2004) 313–332.
[61] S. Koenig, Y. Liu, Terrain coverage with ant robots: a simulation study, in: AGENTS'01, 2001.
[62] S. Koenig, B. Szymanski, Y. Liu, Efficient and inefficient ant coverage methods, Annals of Mathematics and Artificial Intelligence 31 (2001) 41–76.
[63] B. Szymanski, S. Koenig, The complexity of node counting on undirected graphs, Technical Report, Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY.
[64] I. Wagner, M. Lindenbaum, A. Bruckstein, Efficiently searching a graph by a smell-oriented vertex process, Annals of Mathematics and Artificial Intelligence 24 (1998) 211–223.
[65] S.B. Thrun, Efficient exploration in reinforcement learning—technical report cmu-cs-92-102, Technical report, Carnegie Mellon University, 1992.
[66] G. Dudek, M. Jenkin, E. Milios, D. Wilkes, Robotic exploration as graph construction, IEEE Transactions on Robotics and Automation 7 (1991) 859–865.
[67] A. Broder, A. Karlin, P. Raghavan, E. Upfal, Trading space for time in undirected $s - t$ connectivity, in: ACM Symposium on Theory of Computing, STOC, 1989, pp. 543–549.
[68] L. Lovasz, Random walks on graphs: a survey, Combinatorics 2 (1996) 353–398. Paul Erdos is Eighty.
[69] J. Jonasson, O. Schramm, On the cover time of planar graphs, Electronic Communications in Probability 5 (2000) 85–90. electronic.
[70] M. Abramowitz, I. Stegun, Handbook of Mathematical Functions, in: Applied Mathematics Series, National Bureau of Standards, 1964, p. 55.