

SWARM ROBOTICS FOR A DYNAMIC CLEANING PROBLEM

Yaniv Altshuler and Alfred M. Bruckstein

Computer Science Department
Technion, Haifa 32000 Israel
{yanival, freddy}@cs.technion.ac.il

Israel A. Wagner

IBM Haifa Labs
MATAM, Haifa 31905 Israel
wagner@il.ibm.com

ABSTRACT

Several recent works considered multi agents robotics in static environments (e.g. [2], [4], [5] and others). In this work we examine ways of operating in dynamic environments, in which changes may take place regardless of the agents' activity. The work focuses on a dynamic variant of the known *Cooperative Cleaners* problem (described and analyzed in [2]). This problem assumes a grid, part of which is "dirty", when the "dirty" part is a connected region of the grid. On this dirty region several agents move, each having the ability to "clean" the place it is located in. The dynamic variant of the problem involves a deterministic evolution of the environment, simulating a spreading *contamination*, or *fire*. A cleaning protocol for the problem is presented, as well as several analytic bounds for it. In addition, the work contains simulative results for the proposed protocol.

1. INTRODUCTION

Significant research effort has been invested during the last few years in design and simulation of multi-agents systems ([7], [8], [9], [10]). Unfortunately, the geometrical theory of such multi-agents systems is far from being satisfactory, as pointed out in [11] and many other papers.

In this work we will examine a problem in which the agents must work in dynamic environments — an environment in which changes may take place regardless of the agents' activity. This work focuses on a dynamic variant of the known *Cooperative Cleaners* problem [2]. This problem assumes a grid, part of which is 'dirty', where the 'dirty' part is a connected region of the grid. On this dirty grid region several agents move, each having the ability to 'clean' the place ('tile', 'pixel' or 'square') it is located in. The dynamic variant of the problem (described in section 2) involves a deterministic evolution of the environment, simulating a spreading *contamination* (or spreading *fire*).

In the spirit of [12] we consider simple robots with only a bounded amount of memory (i.e. a *finite-state-machine*).

This work contains a cleaning protocol for the problem (section 3) as well as several analytic bounds for the time it

takes agents which use this protocol to clean the entire grid (a summary of the results is presented in section 4 while a full proof appears in section 5). In addition, the work contains simulation results for the protocol (section 6).

As a motivator, we present a method of constructing an unsolvable case for each pair of values of k (number of agents) and d (the speed in which the environment changes), regardless of the cleaning protocol employed. Section 7 contains a discussion regarding the problem and details about an extended ongoing research.

2. THE DYNAMIC COOPERATIVE CLEANERS PROBLEM

Let us assume that the time is discrete. Let G be a two dimensional grid, whose vertices have a binary property of '*contamination*'. Let $cont_t(v)$ state the contamination state of the vertex v in time t , taking either the value "on" or "off". Let F_t be the dirty 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. A discussion regarding simple connectivity appears in section 3.3.

Let a group of k agents that can move across the grid G (moving from a vertex to its neighbor in one time step) be placed in time t_0 on F_0 (we focus on the cleaning problem, and not on the discovery problem).

Each agent is equipped with a sensor capable of telling the condition of the square it is currently located in, as well as the condition of the squares in the $8-Neighbors$ group of the this square. An agent is also aware of other agents which are located in its square, and all the agents agree on a common "north". Each square can contain any number of agents simultaneously.

When an agent moves to a vertex v , it has the possibility of causing $cont(v)$ to become *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 connected component.

Every d time steps the contamination spreads. That is, if $t = nd$ for some positive integer n , then $(\forall v \in F_t, \forall u \in 4-Neighbors(v) : cont_{t+1}(u) = on)$.

The agents' goal is to clean G by eliminating the contamination entirely, so that $(\exists t_{success} : F_{t_{success}} = \emptyset)$. In addition, it is desired that this $t_{success}$ will be minimal.

In this work we demand that there is no central control and that the system is fully 'de-centralized' (i.e. all agents are identical and no explicit communication is allowed).

3. THE CLEANING PROTOCOL

For solving the *Dynamic Cooperative Cleaners* problem we suggest the **SWEEP** cleaning protocol, which can be described as follows. Generalizing an idea from computer graphics (presented in [13]), we preserve the connectivity of the *contaminated* region by preventing the agents from cleaning *critical points* — points which disconnect the graph of contaminated grid points (see section 5.1). This ensures that the agents stop only upon completing their mission. 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 complete the mission, if possible.

At each time step, each agent cleans its current location (assuming this is not a critical point), and moves to its *rightmost* neighbor — a local movement rule, creating the effect of a clockwise traversal of the contaminated shape. As a result, the agents "peel" layers from the shape, while preserving its connectivity, until the shape is cleaned entirely.

3.1. Cleaning protocol - definitions and requirements

Let $r(t) = (\tau_1(t), \tau_2(t), \dots, \tau_k(t))$ denote the locations of the k agents at time t . The requested cleaning protocol is therefore a rule f such that for every agent i , $f(\tau_i(t), Neighborhood(\tau_i(t)), \mathcal{M}_i(t)) \in \mathcal{D}$, where for a square v , $Neighborhood(v)$ denotes the contamination states of v and its 8-*Neighbors*, \mathcal{M}_i is a finite amount of memory for agent i , containing information needed for the protocol (e.g. the last move) and $\mathcal{D} = \{ 'left', 'right', 'up', 'down' \}$.

Let ∂F denote the boundary of F . A square is on the boundary if and only if at least one of its 8-*Neighbors* is not in F , meaning $\partial F = \{ (x, y) \mid (x, y) \in F \wedge 8-Neighbors(x, y) \cap (G \setminus F) \neq \emptyset \}$.

The requested rule f should meet the following goals :

- **Full Cleanness** : $(\exists t_{success} : F_{t_{success}} = \emptyset)$. Notice that this demand is limited to cases where this is possible. Since we cannot know whether completing the mission is possible, we can only demand that when $d \rightarrow \infty$ the agents should achieve this goal.
- **Agreement on Completion** : within a finite time after completing the mission, all the agents must halt.
- **Efficiency** : in time and in agents' memory resources.

The requested rule should also be *fault tolerant*.

3.2. The SWEEP cleaning protocol

The protocol is being used by the agent a_i which in time t is located in $\tau_i(t) = (x, y)$. The term '*rightmost*' means: "starting from the previous boundary point scan the neighbors of (x, y) in a clockwise order until you find another boundary point". The additional information needed for the protocol and its sub-routines is achieved from \mathcal{M}_i and $Neighborhood(x, y)$. The protocol appears in figure 1.

<p>Protocol SWEEP(x, y) :</p> <p>If (not is-critical(x, y)) and $((x, y) \in \partial F)$ and (there are no other agents in (x, y)) then Set $cont(x, y)$ to off; /*Clean current position*/</p> <p>If (x, y) has no contaminated neighbors then STOP;</p> <p>If (there are no other agents in (x, y)) or (the agent has the highest priority among agents in (x, y)) then If $\neg((x, y) \in \partial F)$ and in the previous time step the agent's square was in ∂F then /* Spread had occurred. Search for ∂F */ Move in 90° counterclockwise to the previous movement and skip the rest of the protocol;</p> <p>If $\neg((x, y) \in \partial F)$ and in the previous time step the agent's square was not in ∂F then /* Keep seeking ∂F */ Move on the same direction as in the previous movement and skip the rest of the protocol;</p> <p>If $(x, y) \in \partial F$ then Go to the <i>rightmost</i> neighbor of (x, y);</p> <p>End SWEEP;</p>
<p>Function is-critical(x, y) :</p> <p>If (x, y) has two contaminated squares in its 4-<i>Neighbors</i> which are not connected via a sequence of contaminated squares from its 8-<i>Neighbors</i> then Return TRUE</p> <p>Else Return FALSE;</p> <p>End is-critical;</p>
<p>Function priority(i) :</p> <p>$(x_0, y_0) = \tau_i(t-1)$; $(x_1, y_1) = \tau_i(t)$; Return $(2 \cdot (x_0 - x_1) + (y_0 - y_1))$;</p> <p>End priority;</p>
<p>Procedure INITIALIZE() :</p> <p>Choose a starting point on ∂F, p_0;</p> <p>Put all the agents in p_0;</p> <p>For $(i = 1; i \leq k; i++)$ do Start agent i according to the SWEEP protocol; Wait 2 time steps;</p> <p>End INITIALIZE;</p>

Figure 1. The **SWEEP** cleaning protocol

The connectivity of the region yet to be cleaned, F , is

preserved by allowing an agent to clean only *non-critical* points. This guarantees both full cleanliness 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 handled, albeit slower.

Since we only allow agents to clean points which are in ∂F , we guarantee that no new ‘holes’ will be created. The simple-connectivity of F , if such exists, is thus kept (see section 3.3 for more details).

Note that when several agents are located in the same square, only the last one who exits it cleans it, in order to prevent agents from being ‘thrown’ out of the contaminated region. This problem could also be solved by implementing a ‘contamination seeking mechanism’ (i.e. by applying methods such as suggested in [14]). That solution, however, would have been far less elegant and would have added additional difficulties to the analysis process.

3.3. Dynamic and static cleaning problems

The work presented in [2] contains a cleaning protocol designed for agents working in static environments (in which $d \rightarrow \infty$), called **CLEAN**. The **SWEEP** protocol is very similar to **CLEAN** albeit with a couple of changes.

In the **CLEAN** protocol for static environments (in [2]), p_0 was artificially treated as a *critical point*. This however, was not crucial for the protocol to work but rather just simplified the analysis of it. Since in the dynamic problem p_0 will not necessarily remain in ∂F , it loses its use as a point in which every agent must pass in each of its traversals around F_t . This means that unlike the **CLEAN** protocol, we cannot foretell which point will be the last point to be cleaned, but this however, is not important to us. Thus, unlike the preservation of the *pivot point*, p_0 in the **CLEAN** protocol, when an agent using the **SWEEP** protocol reaches the *pivot point*, it will clean it.

If an agent finds itself in the middle of the contaminated region (i.e. not in ∂F), it ‘turns left’, and moves straight until reaching ∂F . This is required since after a contamination spread, two blocks can merge, capturing an agent inside a contaminated region. Since in the static problem the ‘dirty’ region was not spreading, this problem did not occur.

Notice that unlike static environments, we are not concerned with ‘holes’ in F since should there be such holes, the agents will not enlarge them (due to the nature of the protocol), and the holes will be gradually closed (since the contamination spreads). Thus even if F_0 is not simply connected, the completion of the mission will not be affected, although the cleaning time might become longer.

This however, does not hold for a certain family of initial shapes. A complete discussion regarding a criteria for F_0 which guarantees that new holes will not be created can be found in [3].

4. RESULTS

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 proposed cleaning protocol is important for estimating its efficiency. The following analytic results are presented in rest of this work. Given a contaminated shape F_0 with initial area of S_0 , and k agents employing *any* cleaning protocol, a lower bound for S_t (the area of F in time t), and thus for the cleaning time of the agents, is presented as Theorem 1 in section 5.2. In addition, an upper bound for $t_{SUCCESS}$, the cleaning time of k agents, using the **SWEEP** cleaning protocol, for a given contaminated shape F_0 is presented as Theorem 3 in section 5.3. From this bound, an upper bound for the number of agents needed to apply the **SWEEP** protocol, for a successful cleaning of a given shape F_0 is derived.

5. ANALYSIS

Notice that analyzing the performance of the protocol is quite difficult. First, due to the preservation of the *critical points*, such points can be visited many times by the agents without being cleaned. Second, due to the dynamic nature of the problem the shape of the contaminated region can dramatically change during the cleaning process.

Another difficulty which arises is that of guaranteeing the completion of the cleaning by the agents. We must show that the agents are cleaning the contaminated region faster than the spreading of the last. For example, for any pair of values of d and k , let F_0 be a straight line of length $\lceil \frac{1}{8}d^2k^2 + \frac{1}{2}dk + 1 \rceil$. Then, by applying the lower bound for the cleaning time (presented in Theorem 1), we can see that the size of the shape is continually growing.

5.1. Definitions

Let S_t denote the size of the contaminated region F in time t , namely the number of grid points (or squares) in F_t .

The boundary of the contaminated region F is denoted as ∂F , defined as $\partial F = \{(x, y) \mid (x, y) \in F \wedge (x, y) \text{ has an } 8\text{-Neighbor in } (G \setminus F)\}$. Let d denote the number of time steps between two contamination spreads.

A *path* in F is defined to be a sequence (v_0, v_1, \dots, v_n) of squares in F such that every two consecutive squares are *4-Connected* (meaning that the Manhattan distance between them is 1). A *length* of a *path* is defined to be the number of squares in it.

Let square v be called a *critical point* if $\exists v_1, v_2 \in 4\text{-Neighbors}(v)$ where all the paths connecting v_1 and v_2 , which are included in $8\text{-Neighbors}(v)$, pass through v .

Let c_t denote the circumference of F in time t , defined as follows. Let $C_t = (v_0, v_1, \dots, v_n)$ be the shortest *path* which contains all the squares of ∂F_t and only such squares,

and such that v_0 and v_n are 4-Connected. If there are several different shortest paths then let C_t be an arbitrary one. Notice that C_t may contain several instances of the same square, if this square is a *critical point*. c_t is defined as the length of C_t . An example appears in Figure 2.



Figure 2. The line in the left chart goes through the squares of C_t where the 3 arrowed squares are included twice. The circles in the right chart denote the squares of ∂F_t . Note that while ∂F_t contains 11 squares, C_t contains 14.

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 in any *non-critical point* in ∂F_t .

For some $v \in F_t$ 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).

Let $W(F_t)$ denote the *width* of F_t and be defined as the maximal depth of all the squares in F_t , meaning that $W(F_t) = \max\{w(F_t, v) \mid v \in F_t\}$.

Let F'_t denote the region one could get, 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. Let $'F_t$ denote the region that would be received had a contamination spread occurred in time t (assuming no squares were cleaned in time t).

5.2. Lower bound

Following is a general lower bound for the problem. The completion of the cleaning mission in time t means that $S_t = 0$. By showing that in a specific time t , S_t is larger than zero, we prove that the mission could not be completed until that time, regardless of the nature of the cleaning protocol is utilized by the agents.

Theorem 1 *Using any cleaning protocol, the area of the contaminated region in time t can be recursively bounded by the formula :*

$$S_{t+d} \geq S_t - d \cdot k + (\sqrt{8 \cdot (S_t - d \cdot k) - 4} + 2)$$

Proof : Notice that a lower bound for the cleaning time is in fact an upper bound for the agents' performance. Let us assume that the agents are working in 100% efficiency, meaning, each time step every agent cleans a single square. After $(d-1)$ time steps k agents will thus clean $k \cdot (d-1)$ squares, and thus we know that $S_{d-1} \geq S_0 - (d-1) \cdot k$

In the d -th time step, the agents clean another portion of k squares, but the remaining contaminated squares spreads

and cause new squares to become contaminated. We are interested in the *minimal* number of squares which can become contaminated at this stage. The minimal number of 4-Neighbors of any number of squares is received when the squares are organized in the shape of a digital sphere, as in figure 3.

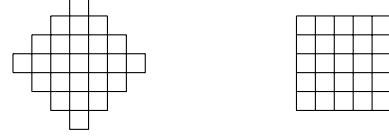


Figure 3. The left chart shows an example of a 'sphere' in the grid. Notice that this sphere has only 16 squares in its 4-Neighbors while the right chart, containing a shape of the same area, has 20 squares in its 4-Neighbors.

Let a denote the diameter of a sphere (in the sphere of figure 3 for example, the diameter is 7). The area of a sphere equals $\frac{a^2+1}{2}$ while the number of squares in its 4-Neighbors (denoted by ΔS) is $(2a+2)$. Thus :

$$\Delta S_{SPHERE} = \sqrt{8 \cdot S_{SPHERE} - 4} + 2$$

After the d -th time steps we thus have the following :

$$S_d \geq S_0 - d \cdot k + (\sqrt{8 \cdot (S_0 - d \cdot k) - 4} + 2)$$

and the Theorem is implied.

□ (End of Theorem 1)

5.3. Upper bound

Lemma 1 *The cardinality of the region's circumference does not increase after it is being traversed by an agent which is using the **SWEEP** protocol, namely : $|\partial F'| \leq |\partial F| - 8$.*

Proof : Note that a traversal around F is a closed, simple, rectilinear polygon. $\partial F'$ was obtained after deleting all the *non-critical* points of ∂F . Going along such polygon, an agent either go straight, make an internal turn ("left" turn, if we assume a clockwise movement), or make an external turn ("right" turn). It is easy to see that a "left" turn increases the traversal length by two, while a "right" turn decreases it by two. Since ∂F is a simple rectilinear polygon, it always has four "right" turns more than "left" ones¹. When *critical* points are met, they are not being cleaned, which means that they exist both in ∂F and in $\partial F'$. Repeating these points, however, does not change the overall size of the set of squares.

□ (End of Lemma 1)

¹This is a simple consequence of the "rotation index" Theorem (see e.g. [6] pp. 396) : If $\alpha : [0, 1] \rightarrow B^2$ is a plane, regular, simple, closed curve, then $\int_0^1 k(s)ds = 2\pi$, where $k(s)$ is the curvature of $\alpha(s)$ and the curve is traversed in the positive direction.

Lemma 2 Every time a region is traversed by an agent using the **SWEEP** protocol, its width is decreased by (at least) one, namely : $W(F'_t) \leq W(F_t) - 1$. A corollary of the latest is that the number of tours around a region F that k agents must accomplish before $F = \emptyset$ is at most $(\frac{W(F)}{k} + 1)$. This holds of course only when no spreads occur.

Proof : By the definition of the depth of a square, it is the shortest path to a *non-critical* square in ∂F . According to the **SWEEP** protocol, after an agent had traversed F , all of the *non-critical* points in ∂F were cleaned. This means that the depth of every internal square was decreased by (at least) one, meaning that the total width of the shape was decreased by (at least) one. Since the **SWEEP** protocol preserves the internal order of the agents, when k agents complete a traversal of F , the width of F is decreased by at least k .

□ (End of Lemma 2)

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 in the same square) is at most $4 \cdot c_t$.

Proof : According to the **SWEEP** protocol, an agent is traversing F at a pace of one square every time step. The only exception occurs when several agents are entering the same square. At each time step the number of agents which are not delayed is at most $\frac{k}{4}$ (see a complete proof in [3]). This means that even if an agent collides with other agents continuously, its average traversal time (amortized for the entire group of agents) is at most 4 times its minimal traversal time (in reality, the agents' traversal time is only slightly more than c_t , although an analytic proof is yet to be found).

□ (End of Lemma 3)

Lemma 4 The length of the circumference of F_t never exceeds twice its cardinality, namely :

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

Proof : ∂F_t is a connected graph and thus it has spanning trees. A protocol for constructing a spanning tree for ∂F_t and a *Depth First Search* (DFS) of it was constructed, such that the path defined by the DFS contains a traversal around F_t (meaning that the DFS is a path which after having several of its squares removed, equals a path which traverses F_t). An example appears in figure 4.

A DFS of a tree can go through each edge at most twice. A tree of $|V|$ vertices contains exactly $(|V| - 1)$ edges. Thus, a DFS of a spanning tree of ∂F_t contains at most $2 \cdot (|\partial F_t| - 1)$ transitions of edges. Since there exists such a spanning tree that contains a 'tour' around F_t then : $c_t \leq 2 \cdot |\partial F_t| - 2$.

□ (End of Lemma 4)

Lemma 5 At any given time, the size of the contaminated region (i.e. $S_t = |F_t|$) is bounded by the following rule :

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

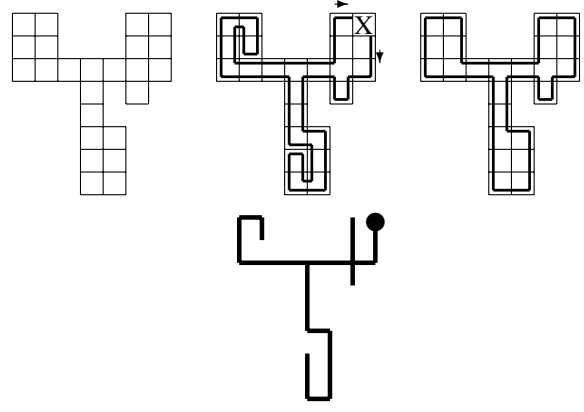


Figure 4. An example of the spanning tree protocol. The left chart represents the contaminated region F_t . The middle chart shows the DFS of ∂F_t according to the **SPAN-DFS** protocol, when the big X marks p_s , the *non-critical* starting point. The right chart shows the traversal path which is contained in this DFS. The drawing on the bottom shows the spanning tree that is created by the protocol and is used for the DFS. See [3] for the **SPAN-DFS** algorithm.

Proof : The only time squares can become contaminated is when spread occurs. When producing an upper bound for S_t , it is safe to disregard the agents' cleaning, since they never produce new contaminated squares. The maximal number of new contaminated squares is achieved when the already contaminated squares have the maximal boundary area possible. This happens when the squares are arranged in the form of a straight line. In such a case, whenever the contamination spreads, the area of the shape is increased by $(c_0 + 2 + 2\iota)$ when ι is a running element of the sequence $(1, 3, 5, 7, \dots)$. Let S_{s_i} denote the area of F_t after the i -th spread. Then for any shape F_t , this is bounded as follows :

$$S_{s_i} \leq S_0 + i(c_0 + 2) + 2(1 + 3 + 5 + \dots + (2i - 1))$$

The last can be simplified to : $S_{s_i} \leq S_0 + i \cdot c_0 + 2(i^2 + i)$. Since a spread occurs every d time steps, t can be written in a form of : $t = i \cdot d + \alpha$ where i is the number of spreads and α — a delay. Since we are only interested in the number of spreads (i.e. i), we can write : $i = \lfloor \frac{t}{d} \rfloor$. By applying this to the previous formula, we get the requested result.

□ (End of Lemma 5)

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

Theorem 2 At any given time, the circumference of the contaminated region F_t is bounded as follows :

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

where

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

Proof : According to Lemma 4, between two spreads, c_t is bounded by twice it's 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).

□ (End of Theorem 2)

Since the initial shape of the contaminated region is unknown, when a spread occurs its width might grow in more than one. This might happen when several 'blocks' merge into one (an example appears in figure 5).

Lemma 6 Assuming no agents are cleaning a shape F , the width of the shape can be bounded as following :

$$W(F_t) \leq \begin{cases} W(F_0), & t \leq d \\ W_{MAX}(t), & t > d \end{cases}$$

where

$$W_{MAX}(t+1) \leq \frac{1 + \sqrt{2S_0 + 2\lfloor \frac{t}{d} \rfloor c_0 + 4(\lfloor \frac{t}{d} \rfloor^2 + \lfloor \frac{t}{d} \rfloor)} - 1}{2}$$

Proof : The maximal width of a shape of X squares is achieved when the squares are arranged as a sphere. The size of a sphere of diameter a is $\frac{a^2+1}{2}$. The width of a sphere of diameter a is $\frac{a+1}{2}$. Thus :

$$W_{SPHERE} = \frac{1 + \sqrt{2 \cdot S_{SPHERE} - 1}}{2}$$

Applying Lemma 5 yields the requested result.

□ (End of Lemma 6)

Figure 5 presents an example for the difficulty of accurately foretelling the new width of a shape after a contamination spread. Notice though that in convex shapes the "merging blocks" phenomenon cannot occur. Thus, after every contamination spread the width can grow by at most 1. This notion appears in Lemma 7.

Lemma 7 If F_t is convex then :

$$W(F_t) \leq W(F_t) + 1$$

$$W(F_{t+1}) \leq W(F_0) + \eta \quad \text{where} \quad \eta \triangleq \left\lfloor \frac{t}{d} \right\rfloor$$

Let $W_{REMOVED}(t)$ denote the decrease in F 's width due to the agents activity until time t (i.e. the number of completed traversals around the contaminated region, multiplied by k). Note that in time step t , each agent 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 4 times the length of this path, then the decrease in the original shapes's width, caused by the agents' cleaning can be bounded as following :

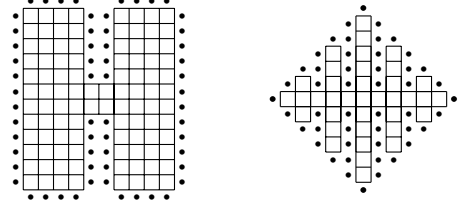


Figure 5. The squared area represents the original shapes while the the dots represent the new squares which are added after a contamination spread. The left chart shows a shape whose width is 2, which spreads to a shape of width 6. The formula in Lemma 6 produces widths of 7 and 9 respectively. The right chart demonstrates the difficulty of foretelling the new width of a shape — while the original width is 1, the shape that is produced after the spread is a sphere, whose width is 7. This example can be easily enlarged in order to produce any requested value for the width of the shape that is created after the spread.

Lemma 8

$$W_{REMOVED}(t+1) \geq \left\lfloor \sum_{i=1}^t \frac{k}{4 \cdot c_i} \right\rfloor$$

A bound over the cleaning time of a given shape F_0 can be produced using the following procedure :

Theorem 3

1. If $(t = \frac{8(|\partial F_0|-1) \cdot (W(F_0)+k)}{k} + 2k)$ is not greater than d , then $t_{SUCCESS} = \lceil t \rceil$. This also holds for static environments, since in such cases $d \rightarrow \infty$.
2. Otherwise ($t > d$) : if F_0 is convex, find the minimal 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

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

3. Otherwise ($t > d$, F_0 is not convex), find the minimal t for which :

$$\begin{aligned} \sum_{i=d+1}^t \frac{1}{S_0 - 1 + 2\lfloor \frac{i}{d} \rfloor^2 + (c_0 + 2)\lfloor \frac{i}{d} \rfloor} &\geq \\ &\geq \alpha + \frac{8}{2k} \sqrt{\beta + 4\left(\left\lfloor \frac{t}{d} \right\rfloor^2 + \left\lfloor \frac{t}{d} \right\rfloor\right)} \end{aligned}$$

where

$$\alpha \triangleq 8 + \frac{8}{2k} - \frac{d - 2k}{|\partial F_0| - 1} \quad \text{and} \quad \beta \triangleq 2S_0 + 2c_0 - 1$$

Proof : In order for F_0 to be cleaned, there should exist a time $t_{SUCCESS}$ in which the width of the shape will be 0. Remembering that $W(F_t)$ is monotonically increasing and disregards the cleaning performed by the agents, and that $W_{REMOVED}(t)$ denotes the decrease in the width of F_0 due to the cleaning of the agents, then we are interested in :

$$W(F_{t_{SUCCESS}}) + k \leq W_{REMOVED}(t_{SUCCESS})$$

The purpose of the “+ k ” is to guarantee the cleaning of the ‘skeleton’ that remains when the width of the shape decreases to zero. In such a time, the agents function as a single agent, and we must guarantee that an additional traversal will be performed. We now apply Lemma 6 (Lemma 7 for convex shapes), and Lemma 8.

We first examine the case of $t_{SUCCESS} \leq d$. Regarding $W(F_t)$, c_t and $W_{REMOVED}(t)$ this means that :

$$\forall i \leq t, W(F_i) \leq W(F_0) \text{ and } \forall i \leq t, c_{i+1} \leq 2(|\partial F_0| - 1) \text{ and } W_{REMOVED}(t) \geq \left\lfloor \sum_{i=1}^t \frac{k}{4 \cdot c_i} \right\rfloor \geq \left\lfloor \frac{k \cdot t}{8(|\partial F_0| - 1)} \right\rfloor.$$

Thus, we are interested in :

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

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

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

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

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

Requesting the last to be greater than the increasing width of the shape, we receive the following, for which we must find the minimal t which satisfy it :

$$\begin{aligned} \sum_{i=d+1}^t \frac{1}{S_0 - 1 + 2\lfloor \frac{i}{d} \rfloor^2 + (c_0 + 2)\lfloor \frac{i}{d} \rfloor} &\geq \\ &\geq \alpha + \frac{8}{2k} \sqrt{\beta + 4\left(\left\lfloor \frac{t}{d} \right\rfloor^2 + \left\lfloor \frac{t}{d} \right\rfloor\right)} \end{aligned}$$

where

$$\alpha \triangleq 8 + \frac{8}{2k} - \frac{d-2k}{|\partial F_0| - 1} \text{ and } \beta \triangleq 2S_0 + 2c_0 - 1$$

In order to find such t , one should add elements to the sum, iteratively, while for every new element, recalculate the value of the right expression. The minimal t for which the above holds is $t_{SUCCESS}$. Performing the above takes $O(t_{SUCCESS})$ time (since every iteration is very cheap).

When F_0 is convex, a tighter bound can be produced. Since for convex shapes $W(F_{i+1}) \leq W(F_0) + \lfloor \frac{i}{d} \rfloor$, using the previous expression for $W_{REMOVED}$ yields :

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

\Downarrow

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

\Downarrow

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

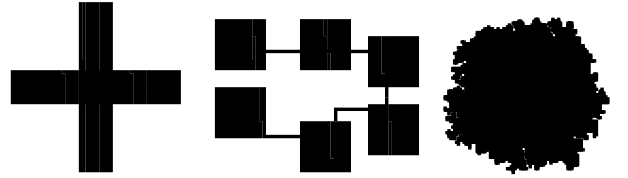
where

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

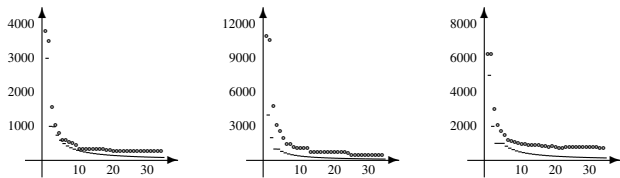
□ (End of Theorem 3)

6. SIMULATIONS

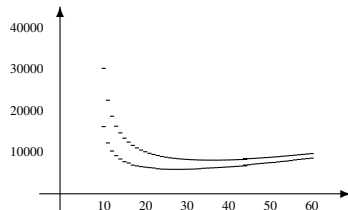
A computer simulation, implementing the **SWEEP** cleaning protocol, was constructed. The simulation examined shapes of various geometric features, and tracked the cleaning time of k agents ($k \in [1, 100]$) which used the **SWEEP** protocol. Following is a sample of the results, including the predictions of the lower and upper bounds. The sample includes a “cross” of size 2960, “rooms” of size 3959, and a random, semi-convex, shape of size 5000, where $d = 1000$:



The lower curves represent the results predicted by the lower bound, while the upper curves represent the actual cleaning time, produced by the simulations performed (the graphs present the cleaning time as a function of the number of agents). The left graph presents the results that were produced by the “cross”, etc’.



The following graph contains the upper bound for the “cross”. Notice that the lower, tighter, curve was produced when taking into account that the “cross” shape is “convex”:



7. DISCUSSION AND CONCLUSION

The paper presents the *Dynamic Cooperative Cleaners* problem, where a group of simple and limited agents must clean a spreading “contaminated” sub-grid. This problem has several “real world” applications. For example, the coordination of fire-fighting units, or an implementation of a distributed anti-virus system for computer networks. Additional applications are distributed search engines, and various military applications.

A cleaning protocol for the problem was presented and analyzed, and several analytic bounds for it, were derived.

While examining the dynamic cooperative cleaners problem, new interesting opportunities for an extended research have emerged. We have already started investigating some of the above, producing more valuable results. Following are various aspects of this ongoing and future research, some of which are to be published elsewhere in the near future.

An explicit upper bound for the cleaning time based on Theorem 3 has been derived, using the Ψ function (see [1]).

In addition, we have recently shown the problem of finding the best cleaning time to be an NP-Hard problem. This increases the motivation of producing analytic bounds for cleaning protocols designed for these problems.

Another important result is the discovery of new geometric features, which are invariants with respect to the agents’ activity and the spreading contamination. Their relevance to the problem was demonstrated by developing an improved upper bound for the cleaning time of the agents.

An interesting aspect is the feasibility question, i.e. foretelling the minimal number of agents required to clean a given shape (regardless of the cleaning time). In addition, developing new cleaning protocols for the problem and producing tighter bounds are also of interest to us. The authors are currently in the stages of developing a tighter bound for the cleaning protocol.

8. REFERENCES

- [1] M. Abramowitz, I.A. Stegun: “Handbook of Mathematical Functions”, National Bureau of Standards Applied Mathematics Series 55, (1964)
- [2] I.A. Wagner, A.M. Bruckstein: “Cooperative Cleaners: A Case of Distributed Ant-Robotics”, “Communications, Computation, Control, and Signal Processing: A Tribute to Thomas Kailath”, Kluwer Academic Publishers, The Netherlands (1997), pp. 289–308
- [3] Y. Altshuler, V. Yanovski: “Dynamic Cooperative Cleaners — Various Remarks”, Technical report, (2005)
- [4] I.A. Wagner, M. Lindenbaum, A.M. Bruckstein: “Efficiently Searching a Graph by a Smell-Oriented Vertex Process”, *Annals of Mathematics and Artificial Intelligence*, Issue 24 (1998), pp. 211–223
- [5] R.C. Arkin, T. Balch: “Cooperative Multi Agent Robotic Systems”, Artificial Intelligence and Mobile Robots, MIT/AAAI Press, Cambridge, MA, (1998)
- [6] M.P. Do-Carmo: “Differential Geometry of Curves and Surfaces”, Prentice-Hall, New-Jersey, (1976)
- [7] R.A. Brooks: “Elephants Don’t Play Chess”, *Designing Autonomous Agents*, P. Maes (ed.), pp. 3–15, MIT press / Elsevier, (1990)
- [8] S.Levi: “Artificial Life - the Quest for a New Creation”, Penguin, (1992)
- [9] S.Sen, M. Sekaran, J. Hale: “Learning to Coordinate Without Sharing Information”, *Proceedings of AAAI-94*, pp. 426–431
- [10] L.Steels: “Cooperation Between Distributed Agents Through Self-Organization”, *Decentralized A.I - Proc. first European Workshop on Modeling Autonomous Agents in Multi-Agents world*, Y.DeMazeau, J.P.Muller (Eds.), pp. 175–196, Elsevier, (1990)
- [11] G.Beni, J.Wang: “Theoretical Problems for the Realization of Distributed Robotic Systems”, *Proc. of 1991 IEEE Internal Conference on Robotics and Automation*, pp. 1914–1920, Sacramento, CA, April (1991)
- [12] V.Breitenberg: *Vehicles*, MIT Press (1984)
- [13] D.Henrich: “Space Efficient Region Filling in Raster Graphics”, *The Visual Computer*, pp. 10:205–215, Springer-Verlag, (1994)
- [14] R. Baeza-yates, R. Schott: “Parallel Searching in the Plane”, *Computational Geometry* 5 pp. 143–154 (1995)