# 16

# COOPERATIVE CLEANERS : A STUDY IN ANT ROBOTICS

**Israel A. Wagner and Alfred M. Bruckstein**

*Computer Science Department*
*Technion - Israel Institute of Technology*
*Haifa, Israel 32000*

*To Tom Kailath, teacher and friend, with appreciation and best wishes for the next sixty great years!*

## ABSTRACT

In the world of living creatures, "simple minded" animals often cooperate to achieve common goals with amazing performance. One can consider this idea in the context of robotics, and suggest models for programming goal-oriented behavior into the members of a group of simple robots lacking global supervision. This can be done by controlling the local interactions between the robot agents, to have them jointly carry out a given mission. As a test case we analyze the problem of many simple robots cooperating to clean the dirty floor of a non-convex region in $\mathbf{Z}^2$, using the dirt on the floor as the main means of inter-robot communication.

## 1 INTRODUCTION

In the world of living creatures, "simple minded" animals like ants or birds cooperate to achieve common goals with surprising performance. It seems that these animals are "programmed" to interact locally in such a way that the desired global behavior is likely to 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 achieve reliability through redundancy. Significant research effort has been invested during the last few years in design and simulation of multi-agent

systems (e.g. [20], [15], [4], [19]). Unfortunately the geometrical theory of such multi-agent systems is far from being satisfactory, as has been pointed out in [2] and many other papers.

Our interest is focused on developing the mathematical tools necessary for the design and analysis of such systems. In a recent report [21] we have shown that a number of agents can arrange themselves equidistantly in a row via a sequence of linear adjustments, based on a simple "local" interaction. The convergence of the configuration to the desired one is exponentially fast. A different way of cooperation between agents is inspired by the behavior of ant-colonies, and is described in [6]. There it was proved that a sequence of ants engaged in deterministic chain pursuit will find the shortest (i.e. straight) path from the ant-hill to the food source, using only local interactions. In [7] we investigate the behavior of a group of agents on $\mathbf{Z}^2$, where each ant-like agent is pursuing his predecessor, according to a discrete biased-random-walk model of pursuit on the integer grid. We proved there that the average paths of such a sequence of a(ge)nts engaged in a chain of probabilistic pursuit converge to the straight line between the origin and destination, and this happens exponentially fast.

In this paper we investigate a more complicated question concerning the behavior of many agents cooperating to explore an unknown area (for purposes of cleaning, painting, etc.), where each robot/agent can only see his near neighborhood, and the only way of inter-robot communication is by leaving traces on the common ground and sensing the traces left by other robots. We present an algorithm for cleaning a dirty area that guarantees task completion (unless *all* robots die) and prove an upper bound on the time complexity of this algorithm. We also show simulation results of the algorithm on several types of regions. These simulations indicate that the precise time depends on the number of robots, their initial locations, and the shape of the region.

In the spirit of [3], we consider simple robots with only a bounded amount of memory (i.e. a *finite-state-machine*). Generalizing an idea from computer graphics, [10], we preserve the connectivity of the "dirty" region by allowing an agent to clean only so called *non-critical* points, points that do not disconnect the graph of dirty grid points. This ensures that the robots will 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. We prove the correctness of the algorithm as well as an upper bound on its running time, and show how our algorithm can be extended for regions with obstacles.

The rest of the paper is organized as follows: in Section 2 the formal problem is defined as well as the aims and assumptions involved. The algorithm is presented in Section 3, where its time-complexity is analyzed, and some simulation examples are shown. Section 4 is devoted to the problem of exploring/cleaning regions with obstacles, and is followed by some examples from our simulation

runs. We conclude the paper with a discussion and by pointing out several connections to related work.

# 2 COOPERATIVE CLEANING: PROBLEM STATEMENT

The problem we address is the following: we are given a region $R_0$ to be explored, which is assumed to be a connected subset of the integer grid, $\mathbf{Z}^2$, and a set of $k$ robots initially located at $\mathbf{r}(0) = (r_1(0), r_2(0), \ldots, r_k(0))$. We want these robots to clean $R_0$, i.e. initially all points of $R_0$ are assumed to be dirty ("off",0) and must be turned clean ("on",1). The system should evolve according to a rule that can be described as

$$(\mathbf{R}(t+1), \mathbf{r}(t+1)) = f\,(\mathbf{R}(t), \mathbf{r}(t))$$

where $\mathbf{R}(t)$ is the *status map* of $R_0$ at time $t$ - indicating the region still to be cleaned by 0's and designating those points that have already been cleaned by 1. Initially all bits are set to indicate the status of all points in $R_0$ as 0.

We seek for a rule $f$ that will fulfill the following aims

1. **Full coverage:** each point in $R_0$ is eventually cleaned.
2. **Agreement on Completion:** After the whole region has been cleaned, there should eventually be a time when all the robots that are still active become *aware* of this fact (and hence can go and take a nap or proceed to the next mission).
3. **Efficiency:** in time and space (memory usage of each robot).
4. **Fault Tolerance:** if some of the robots die and hence stop working before completion, the others should be able to carry on and complete the mission.

We make the following assumptions

1. A robot can move one grid unit at a time, from a grid point to an adjacent one.
2. Each point in $R_0$ represents a room and several robots may occupy the same room at the same time.
3. There is no prior knowledge of the shape of $R_0$ except that it is assumed to be simply connected [1].
4. There is no central control; it is a 'de-centralized' system with no leader (i.e. all agents/robots are identical).

[1] We shall later show how to deal with non-simple regions too.

5. A robot can see and sense the status of his 8-neighborhood. (i.e. the $3 \times 3$ square around its current location).

# 3 A MULTI-AGENT CLEANING ALGORITHM

To solve the cooperative cleaning problem, we suggest the protocol **CLEAN** (see Figure 16.1), to be programmed into and executed by each robot at every (discrete) point of time.

Each robot has, by assumption, a bounded amount of memory (i.e. it is a finite automaton). The connectivity of the region still to be cleaned, $R$, is preserved by allowing a robot to clean only *non-critical* points, i.e. points that do not disconnect the current graph of dirty grid points. This way, it is also guaranteed that the robots will 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 agents will take over their responsibilities and complete the mission. The "pivot" is a special point on the boundary of $R$, $\partial R$ ($\partial R$ is defined as the set $\{(u,v) \,|(u,v) \in R$ and $(u,v)$ has an 8-neighbor in $\mathbf{Z}^2 \setminus R\}$ ), and all robots are assumed to be initially located there. This point (denoted $p_0$) is artificially set to be critical during the execution, hence it is also guaranteed to be the last point cleaned. Completion of the mission can therefore be concluded from the fact that all (working) robots are back at $p_0$ with no dirty neighbors to go to, thereby reporting on completion of their individual missions. Note that the pivot is not necessary for the algorithm to work well - it just makes life easier for the user since one then knows where to find the robots after cleaning has been completed. If we do not start with all robots at the pivot and force $p_0$ to be critical, the location of the robots upon completion will generally not be known in advance. We denote the initial dirty region by $R_0$ and the remaining dirty region by $R$; our aim is to finish with $R = \phi$.

The introduction of the notion of critical points makes time-complexity analysis significantly harder since a critical point may be visited several times before its cleaning. We conjecture that the algorithm proposed is efficient, and additional robots will speed up the cleaning process only up to a limit. In order to give this argument a rigorous form, we need some definitions. We denote the area of $R$ by $a(R)$. In the "grid" context, this area was defined as the number of grid points in $R$. Actually, $R$ defines a dichotomy of $\mathbf{Z}^2$ into $R$ and $\overline{R} = \mathbf{Z}^2 \setminus R$. The *border* of $R$, denoted $\partial R$, was defined as the set of points in $R$ which have 8-neighbors in $\overline{R}$. We define the circumference of $R$, $c(R)$, to be the area of $\partial R$. A *path* in $R$ will be defined as a sequence $(p_1, p_2, \ldots, p_m)$ of points in $R$ such that any two consecutive points are 4-connected in $R$. (Two points are 4-connected if the Manhattan distance between them is one). We now formalize a

notion of $R$'s "fatness" - i.e. its maximal width - as following. Consider $u \in R$, a point in $R$. A *string* of $u$ is a simple path in $R$ that starts at $u$ and ends at a non-critical border point. The *depth* of $u$, denoted by $w(R,u)$, will be defined as the shortest string of $u$ (unless $u$ is itself critical - then its depth will be defined to be zero). The *length* of a string is defined as the number of points in it. The *total width* of $R$, denoted $w(R)$, will be defined as the maximal depth among all points of $R$: $w(R) = \max_{u \in R} \{w(R,u)\}$. In the **CLEAN** algorithm, the connectivity of the dirty shape is always preserved, since a point is never erased as long as it is a critical one. We shall assume that some (arbitrary) point $p_0$ on the boundary of $R$ is a starting point for our agents, and we shall call it a *pivot*. The longest in-region distance between $p_0$ and any other point in $R$ will be referred to as $l(R)$, the *length* of $R$:

$$l(R) = \max_{v \in R} \{d_R(p_0, v)\}$$

where $d_R(x,y)$ is the distance (shortest path within $R$) between $x$ and $y$. See Figure 16.3 for an illustration of the above definitions.

With these concepts defined, we have the following results.

**Lemma 16.1** *If $a(R) > 0$ then $w(R) \geq 1$, that is: any finite simple region has at least two non-critical points on its boundary.*

**Proof:** The boundary $\partial R$ is a connected graph, hence it has a spanning tree. In such a tree there are at least two vertices with degree 1; these vertices are necessarily not critical in $R$, since any boundary point which is critical in $R$ is also critical in $\partial R$. □

**Theorem 16.1** *A group of robots executing the* **CLEAN** *protocol will eventually clean a simply connected region and stop together at $p_0$, the pivot set by the "INITIALIZE" phase of the algorithm.*

**Proof:** While $R$ has not yet been cleaned, $a(R) > 0$ and hence, by Lemma 16.1 there is a non-critical point on $\partial R$. Since the robots obey the **CLEAN** rule, at least one of them will arrive to a non-critical point once in a period of $a(R)$ (since $c(R) \leq a(R)$), and erase this point from $R$, thus reducing $a(R)$ by 1. We conclude that after no more than $a^2(R)$ units of time we shall have $a(R) = 0$. The fact that all robots will meet at the same point is implied by the following two rules that are implemented in the **CLEAN** algorithm:

- **rule 1:** $R$ is always being kept connected. (We never clean a room that has no clean neighbor and never clean a critical room either).
- **rule 2:** The pivot $p_0$ is cleaned only when no other dirty points are left in $R$.

This completes the proof of Theorem 16.1. □

Eventually, as the agents progress in their job, each point becomes non-critical and is cleaned. The point last cleaned is the pivot $p_0$, since we (artificially) keep it critical until all other points have been cleaned. Let us denote the $j$'th robot time of its $i$'th visit to $p_0$ by $\tau_j^i$. The $i$'th *tour* of the boundary by robot $j$, denoted $T_j^i$, is the path traversed by robot $j$ between two consecutive visits at $p_0$, namely:

$$T_j^i = [r_j(\tau_j^i), r_j(\tau_j^i + 1), r_j(\tau_j^i + 2), \ldots, r_j(\tau_j^{i+1})].$$

The path of robot $j$ can then be decomposed into a series of tours:

$$\text{path}(j) = T_j^1, T_j^2, \ldots, T_j^M$$

where $r_j(t)$ is the location of $j$ at time $t$, $\tau_j^i$ is the start-time of $T_j^i$, the $i$'th tour of the boundary by $r_j$, and $M$ is the total number of tours. In the **CLEAN** algorithm, the inter-robot order never changes; that is - there is exactly one visit of $j$ at $p_0$ between two consecutive visits of $i$ at this point, for any $i, j$ such that $1 \leq i, j \leq k$. Note that due to Lemma 16.1, and the order-preservation property of the **CLEAN** algorithm, the total number of tours is the same for all robots.

Hence we can order the tours by the order they visit the pivot $p_0$:

$$T_1^1, T_2^1, \ldots, T_k^1, T_1^2, T_2^2, \ldots, T_k^M$$

The next Lemma states that tour-cardinalities are non-increasing.

**Lemma 16.2** *The cardinality of the set of points visited during a tour cannot be larger than the preceding one, namely*

$$v(T_{j+1}^m) \leq v(T_j^m) - 8$$

*where $v(T)$ denotes the number of vertices in a tour $T$.*

**Proof:** First we note that each tour is a simple, closed, rectilinear polygon. $T_{j+1}^m$, the $m$'th tour of robot $r_j$ is created by tracking the previous tour $T_j^m$, while deleting any non-critical points along the way. Going along such a polygon, we either go straight, turn right or turn left. For the sake of simplicity, let us denote $T_{j+1}^m$ by $T'$ and $T_j^m$ by $T$. It is easy to see (see Figure 16.4(a),(b)) that a right turn increases the tour-length by two and a left turn decreases it by two. (Going straight has no effect). But the tour is a simple rectilinear polygon, hence it always has four "left" turns more than "right" turns (This is a simple consequence of the "rotation index" Theorem (see, e.g. [8] pp. 396): If $\alpha : [0, 1] \to R^2$ 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 (i.e. with the inside to the left of the walker)). The only exception occurs when one or more points along the tour are critical (see Figure 16.4(c)). In such a case, the critical points are just repeated in $V(T')$ but do not increase the overall size of the set. □

Note that the length of the tour may increase but its cardinality will never exceed that of the first one, namely $|\partial R|$.

Next we establish a link between the cardinality of a tour and its length:

**Lemma 16.3** *The length of a tour never exceeds four times its cardinality, that is:*

$$c(T) \leq 4v(T) \tag{16.1}$$

*where $c(T)$ is the length of tour $T$ and $v(T)$ is the number of points in it.*

**Proof:** The tour $T = (V, E)$ is a directed cyclic path traversing the vertices of $V = V(\partial R)$, the border of $R$, along the edges $E = E(\partial R)$ on the boundary. Observing that the maximal degree in $R$ (and hence in $\partial R$) is 4, and that no edge in $T$ is traversed more than twice, one can easily see that

$$c(T) \leq 2\,|E(T)| \leq \sum_{u \in V} d(u) \leq 4v(T)$$

□

The conclusion from Lemma 16.2 and Lemma 16.3 is that a tour length never exceeds twice the original circumference of the region. We next proceed to show that the total number of tours is bounded above. Let us define $w(T_j^m)$, the width of the $m$'th tour of robot $j$, as the width of the region surrounded by this tour. Then we have

**Lemma 16.4** *The width of robots' tours is monotonically decreasing, that is:*

$$\forall j, m \quad 1 \leq j \leq k, \quad 1 \leq m \leq M: \quad w(T_j^m) \leq \max\left\{0, w(T_{j-1}^m) - 1\right\}$$

**Proof:** By definition, $w(T_i^j)$ is the longest distance from an internal point of $T_j^m$ to a non-critical point on its boundary. But, according to the **CLEAN**ing rule, robot $j$ has cleaned all non-critical points from the boundary of $T_{j-1}^m$ - hence its width has decreased by one, the only exception being when $T_{j-1}^m$ is already equal to one. □

From Lemma 16.4 it clearly results that the number of tours, $M$, is bounded:

**Corollary 16.1**

$$M \leq \frac{w(R_0)}{k} + 1$$

**Proof:** According to Lemma 16.4 there are at most $\frac{w}{k}$ tours before the width reduces to 1. Once $w = 1$ all that remains from $R_0$ is a skeleton that can clearly be cleaned in a single tour. □

A relation between the length and duration of each tour is stated next.

**Lemma 16.5** *The time it takes robot $j$ to make its $i$'th tour is bounded above by 1.5 times the geometric length of the tour, namely:*

$$\tau_j^{i+1} - \tau_j^i = 1.5 \cdot c(T_j^i)$$

**Proof:** With the **CLEAN** procedure the robot follows its tour step-by-step, the only exception being when more than one robot enter the same point at the same time, going in the same direction. But, this cannot happen with more than 4 robots (since any point has no more than 4 neighbors), and, such collisions are resolved by releasing the 4 robots at 4 consecutive time points. The times that the four robots have to wait in this location are $0, 1, 2$ and $3$ units of time, respectively. Hence the ratio of the "efficient" time to the total time is $\frac{1+2+3}{9} = \frac{2}{3}$, so the total touring-time of a robot cannot exceed 1.5 times the length of the tour. □

We are now ready to prove the main timing theorem:

**Theorem 16.2** *Assume that $k$ robots start at some boundary point $p_0$ of a simple connected region $R_0$ and work according to the* **CLEAN** *algorithm, and denote by $t_k$ the time needed for this group to clean $R_0$. Then it holds that:*

$$\max\left\{2k, \left\lceil \frac{a}{k} \right\rceil, 2l\right\} \leq t_k \leq 2k + 6c\left(\frac{w}{k} + 1\right) \tag{16.2}$$

*where a,c,l and w denote the area, circumference, length and width of of $R_0$, respectively.*

**Proof :**

1. The lower bound is quite obvious - the left term $2k$ is the time needed to release the $k$ robots from the pivot, $\frac{a}{k}$ is a lower bound on the time necessary to cover the region if the robots were optimally located at the beginning. The right term, $2l$, comes from the observation that at least one robot should visit (and return from) any point of $R_0$, including the one farthest from $p_0$, to which the distance is $l$.

2. According to Lemma 16.2 all robots stop simultaneously at $p_0$ after completion of $M$ tours. Hence, we can estimate the total stopping time as the stopping time for any of the robots, say robot 1:

$$t_k = \tau_1^M = \tau_1^1 + \sum_{m=1}^{M} (\tau_1^{m+1} - \tau_1^m)$$

by Lemma 16.4, Corollary 16.1 and Lemma 16.5, we get:

$$t_k \leq M \cdot \max_{m=1,\ldots,M} \{c(T_1^m)\} \leq 6 \cdot \left(\frac{w(R)}{k} + 1\right) \cdot c(R) = 6\left(\frac{wc}{k} + c\right)$$

3. The additional term of $2k$ stands for the time of release - since all robots are initially concentrated in one point and they keep a separation of two time units between them, we need $2k$ units of time before all $k$ robots become operational.

□

Using the above theorem we can bound the *speedup ratio*, defined as

$$t_k(R_0)/a(R_0),$$

which expresses the benefit of using $k$ robots for a cleaning mission:

**Corollary 16.2**

$$\max\left\{\frac{2k}{a}, \frac{1}{k}, \frac{2l}{a}\right\} \leq \frac{t_k}{a} \leq \frac{2k}{a} + 6\left(\frac{wc}{ka} + \frac{c}{a}\right) \tag{16.3}$$

*where a,w,c,l and k are as in Theorem 16.1.*

An interesting result of Corollary 16.2 is that when $a >> k >> w$, i.e. the number of robots is large relative to the width but small compared to the area, then the speedup is bounded below by twice the ratio of the length and area, and bounded above by $6\frac{c}{a}$, three times the ratio of the area and circumference of $R_0$. Note here the similarity to the ratio $\frac{c}{\sqrt{a}}$, known as the *shape-factor.*

Another conclusion is that when we scale up the region by a factor of $n$, the area increases as $n^2$ but the width, length and circumference all increase as $n$ so we get

**Corollary 16.3**

$$\text{as } n \to \infty, \qquad L_\infty \leq \frac{t_k}{a} \leq U_\infty$$

*where*

$$L_\infty = \frac{1}{k}, \qquad U_\infty = 3\frac{c_0 w_0}{k a_0}$$

*and $a = a_0 n^2, c = c_0 n, w = w_0 n$ are the scaled area, circumference and width, respectively.*

# 4 REGIONS WITH OBSTACLES

So far we have only dealt with simply connected regions, i.e. - regions with no "holes". In the case of a (connected) region with obstacles (i.e. holes) the simple **CLEAN**ing algorithm will not work, due to the following "cycle" problem: eventually, each obstacle will be surrounded by critical points, and there will be a time when *all* boundary points of $R$ will be critical, contrary to the statement of Lemma 16.1. (We shall call such a situation *useless*, as opposed to the *useful* state when some points are cleaned during a tour). As a cure to this problem, we suggest to add an "odoring" feature to our cleaners; that is, a robot will be able to label a point on the floor by a small amount of "perfume". (This action may remind one of the pheromones left by ants during their walks). These labeled points will designate the external boundary of the dirty region. Upon getting to the useless state (detected by each robot due to no cleaning between two consecutive visits to the pivot) a robot will continue to traverse the boundary, but will now look for a point which has a "mixed" boundary - that is, one that has odor on one side but no odor on the other. The robot will clean this point (despite its "criticality" - it is not really critical since it is necessarily part of a cycle around an obstacle) and then continue as in a useful state. This will open one cycle, hence, if there are $s$ obstacles, we will need $s/k$ such tours before the region is completely clean. (See Figure 16.2 for the modified algorithm). On the other hand, Lemma 16.2 no longer holds (see Figure 16.6) since the boundary area can increase with time. However the boundary is always bounded above by the area. We now make the following

**Conjecture 16.1** *Assume that $k$ robots start at some boundary point of a non-simple connected region $R$ with $s$ obstacles in it, and work according to the* **CLEAN-WITH-OBSTACLES** *algorithm, and denote by $t_k$ the time needed for this group to clean $R$. Then it holds that:*

$$\max\left\{2k, \left\lceil\frac{a}{k}\right\rceil, 2l\right\} \leq t_k \leq 2k + 1.5 \cdot a\left(\frac{w}{k} + \left\lceil\frac{s}{k}\right\rceil\right) \qquad (16.4)$$

*where a,c,l and w denote the free area, circumference, length and width of of $R$, respectively, and $s$ is the number of obstacles in $R$.*

# 5 SIMULATION EXAMPLES

A simple motion rule is the core of the cleaning algorithm, as shown in Figures 16.1 and 16.2. Each robot checks if his location is critical. If not - he cleans this location. Then, if the robot is the only one at the current point, it looks around and goes to the rightmost free grid point that is closer than others to the border of the region. If a robot with higher priority occupies the same location, the robot waits.

We ran the algorithm on several shapes of regions and for numbers of robots varying from 1 to 20. See Figure 16.5 - 16.6 for some examples of the evolution

of the layout with time. The gray level of each pixel designates the index of the robot that actually cleaned this point. The right side of Figure 16.5 shows the same region and number of robots as in the left side of this Figure, but with randomly chosen initial locations at the corners. It can be seen that the dirty region is cleaned in a similar way. It should be said here that all the theory we developed in the previous sections (up to a small additive constant) applies to the case where the robots are initially located in randomly selected points on the boundary of $R_0$ (rather than starting from $p_0$). Figure 16.6 shows the evolution of the **CLEAN-WITH-OBSTACLES** algorithm for the same shapes with four additional obstacles in each, with 2 robots (left) and 10 robots (right).

Figure 16.7 summarizes the timing results of many simulations, plotting the time (normalized by area) vs. number of robots, compared to the theoretical bounds. In Figure 16.8 we show the results for the same figures with additional obstacles, together with the conjectured theoretical bounds.

# 6 RELATED WORK AND DISCUSSION

Our cooperative **CLEAN**ing algorithm can be considered as a case of social behavior in the sense of [17], where one induces multi-agent cooperation by forcing the agents to obey some simple "social" guidelines. This raises the question what happens if one robot malfunctions. We have shown that if less than $k$ robots stop, the others will take over their responsibilities. But what if some robots start to cheat ? Such adversaries will have catastrophic consequences, since a crazy robot may clean a critical point and disconnect the dirty region.

Another question of interest is the resolution of collisions between robots. In the **CLEAN** algorithm we resolve such a problem by giving each robot a priority measure depending on his previous location. But it is an interesting open question whether a coin flipping is better here.

The cleaning problem discussed is related to the geometric problem of pocket-machining, see [9] for details. An interesting problem of cleaning and maintenance of a system of pipes by an autonomous robot is discussed in [14]. The importance of cleaning hazardous waste by robots is described in [11].

Our approach is that cleaning is always done at the boundary. It is possible that a better efficiency will be achieved using other approaches:

1. Given that several neighbors are dirty, visit the non-critical ones first (even if not on the boundary). This approach is quite efficient for one robot, but can be a mess for several robots.

2. Once entering a large "room" (that is - upon passing from a critical area to a non-critical one) - designate the entrance by a special type of token,

so that other robots will enter only in case there is no other work to do. This approach guarantees that the robots will be distributed between the large rooms of the $R$-configuration. This is attractive if the region has such rooms of quite similar areas.

3. Quite a different idea is to divide the work into two phases - the first one of "distribution" - the robots locate themselves uniformly around the area. Then, in the second phase, each robot cleans around his "center". If the distribution is appropriate, there will be minimum of interactions between robots in the second phase.

We use the dirt on the floor as a means of inter-robot communication, but other ways for communication between agents have been suggested. One is to use heat trails for this end, as was reported in [16]. In [20], self-organization is achieved among a group of lunar robots that have to explore an unknown region, and bring special rock-samples back to the mother-spaceship, by programming each robot to drop a crumb at each point he visits and walk around at random with a bias toward the negative gradient of crumb concentration.

Another question of interest is how to guarantee covering, at least with high probability, without using any external signs, using only the inter-robot collisions as an indicator for a good direction to proceed.

It is of interest to notice here that an off-line version of the problem, that is: finding the shortest path that visits all grid points in $R$, where $R$ is completely known in advance, is NP-hard even for a single robot. It is a corollary of the fact that Hamilton path in a non-simple grid-graph is NP-complete [12].

In summary, we would like to cite a statement made by a scientist after watching an ant making his laborious way across a wind-and-wave-molded beach[18]:

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

Such a point of view, as well as the results of our simulations and analysis, make us believe that even simple, ant-like creatures, yield very interesting, adaptive and quite efficient goal oriented behavior.

*

---

Procedure **INITIALIZE** ($\mathbf{r}_1$, $\mathbf{r}_2$,...,$\mathbf{r}_k$,$\mathbf{p}_0$):
I.1) Choose a pivot $p_0 \in \partial R_0$;
I.2) Locate the robots at $p_0$;
I.3) For($t = 1$; $t \leq k$; $t++$)
    I.3.1) Let robot $r_t$ start working at $p_0$
        according to the **CLEAN** protocol;
    I.3.2) Wait 2 units of time;
end **INITIALIZE**.

---

Protocol **CLEAN(r,x,y)**:
A) if not(**is-critical**$(x, y)$) then /* can I clean my current location ? */
        Set $R(x, y)$ to 1;
B) if $(x, y)$ has no dirty neighbors then STOP.
C) if [there are no other robots at $(x, y)$] or /* plan the next move */
    [**priority**(r) is higher than the priorities
        of the other robots at $(x, y)$]
            then go to the rightmost neighbor of $(x, y)$ on $\partial R$.
end **CLEAN**.

---

Function **is-critical(x,y)**: /* criticality test */
if [$(x, y)$ has two dirty 4-neighbors which are not connected
    via the 8-neighborhood of $(x, y)$ (excluding $(x, y)$ itself)]
    or [$(x, y) = p_0$]
        then return(TRUE);
            else return(FALSE);
end **is-critical**.

---

Function **priority(r)**: /* priority measure */
$(x_0, y_0) :=$ $r$'s previous location;
$(x_1, y_1) :=$ $r$'s current location;
    return$(2 \cdot (x_0 - x_1) + (y_0 - y_1))$;
end **priority**.

---

**Figure 16.1** The **CLEAN** protocol for robot $r$ currently at $(x, y)$, and its sub-functions. In step $C$, *rightmost* means: starting from the previous boundary point sweep the neighbors of $(x, y)$ in a clockwise order until you find another boundary point

---

Procedure **INITIALIZE** ($\mathbf{r}_1$, $\mathbf{r}_2$,...,$\mathbf{r}_k$,$\mathbf{p}_0$):
I.1) Choose a pivot $p_0 \in \partial R_0$;
I.2) Locate the robots at $p_0$;
I.3) For each $r$, status($r$):=**useful**;
I.4) For($t = 1$; $t \leq k$; $t++$)
    I.4.1) Let robot $r_t$ start working at $p_0$
        according to the **CLEAN-WITH-OBSTACLES** protocol;
    I.4.2) Wait 2 units of time;
end **INITIALIZE**.

---

Protocol **CLEAN-WITH-OBSTACLES(r,x,y)**:
A) if not(**is-critical**$(x, y)$)  /* can I clean my current location ? */
    or [status($r$)=**useless** and
        there is odor on only one side of $(x, y)$]
            then
                Set $R(x, y)$ to 1;
                Set odor$(x, y)$ to 1;
B) if $(x, y)$ has no dirty neighbors then STOP.
C) if [there are no other robots at $(x, y)$] or  /* plan the next move */
    [**priority**(r) is higher than the priorities
        of the other robots at $(x, y)$]
            then go to the rightmost neighbor of $(x, y)$ on $\partial R$.
D) if [$(x, y) = p_0$ and no point has been cleaned by $r$
    since previous visit to $p_0$]
        then status($r$):=**useless**;
end **CLEAN-WITH-OBSTACLES**.

---

**Figure 16.2** The **CLEAN-WITH-OBSTACLES** protocol. Note the use of *odor* to identify cycles. The **priority** and **is-critical** functions are the same as in **CLEAN** algorithm.

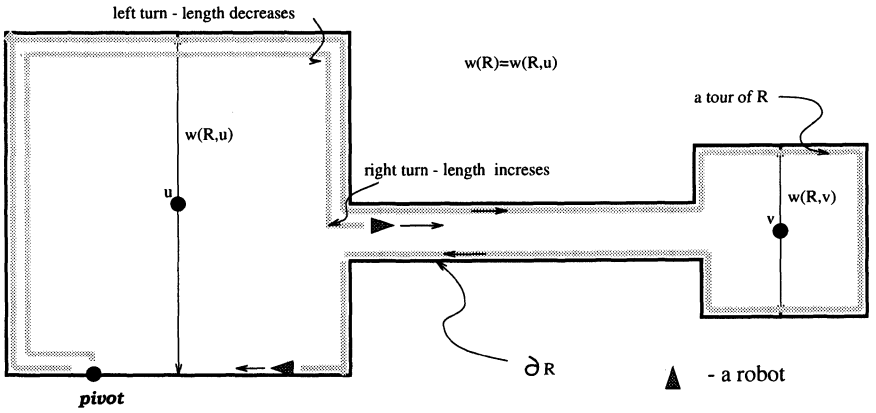


**Figure 16.3** An illustration of the definitions involved in Theorem 16.1.

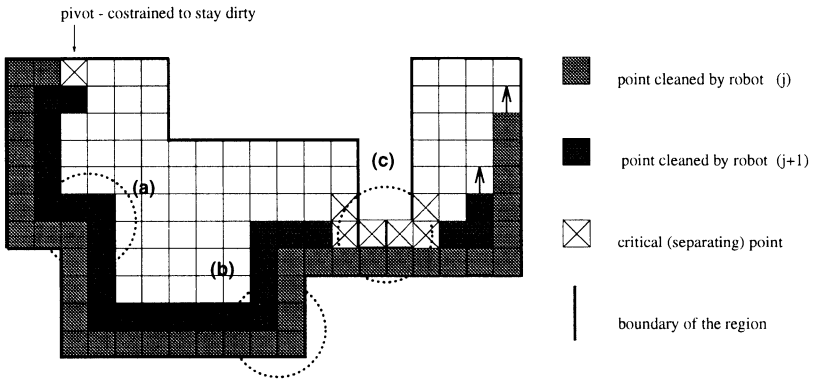


**Figure 16.4** The effect of corners on tour-size may be either an increase (a,c) or decrease (b)

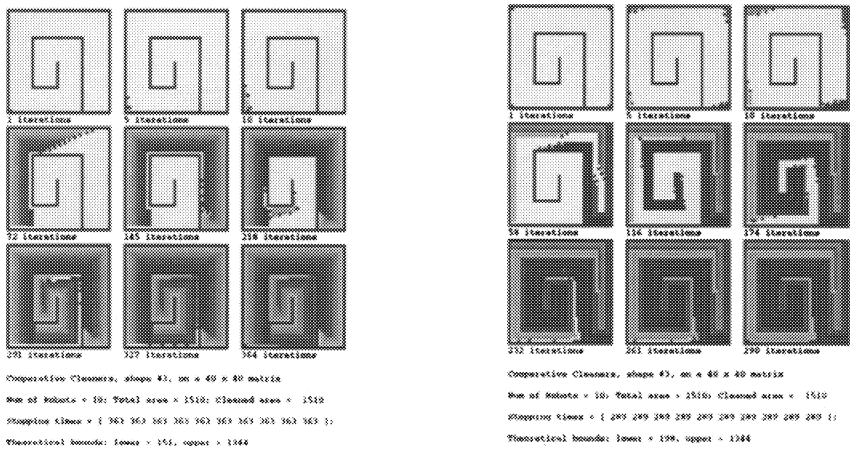

**Figure 16.5** Cooperative cleaners: maze, 10 robots (left), and 10 robots with random initial locations on the boundary (right).

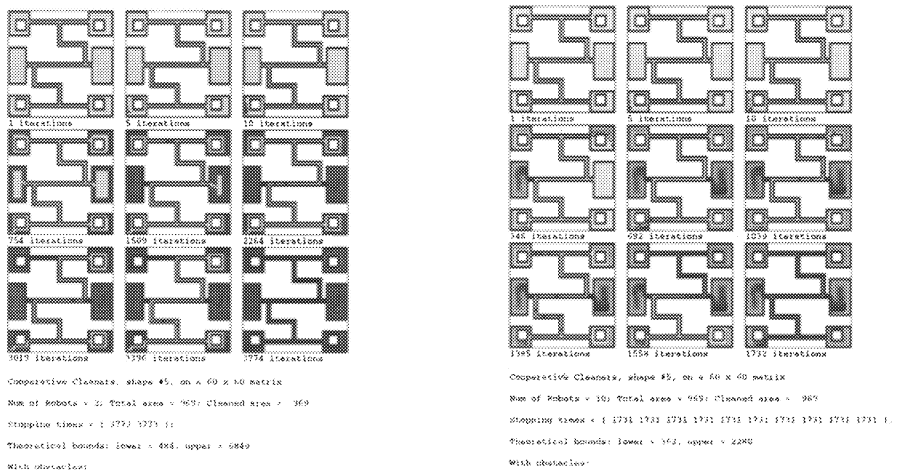

**Figure 16.6** Cooperative cleaners: 2 robots, 6 rooms with obstacles (left), 10 robots, 6 rooms with obstacles (right).

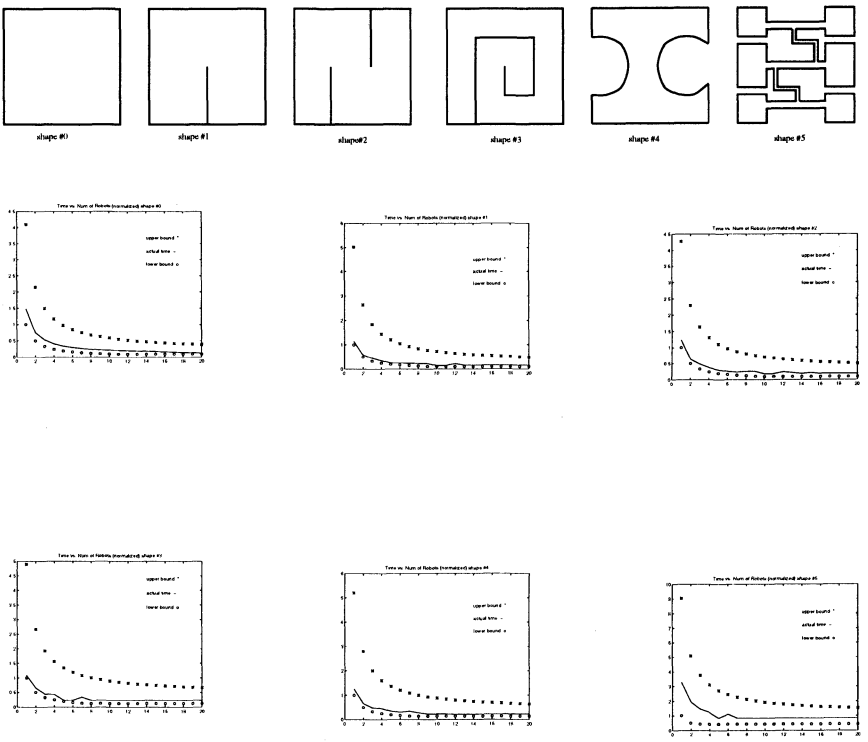

**Figure 16.7** Various shapes tested in **CLEAN** simulations and the normalized time $\left(\frac{t_k}{a}\right)$ vs. number of robots ($k$) for each shape, together with the lower and upper bounds according to Theorem 16.2

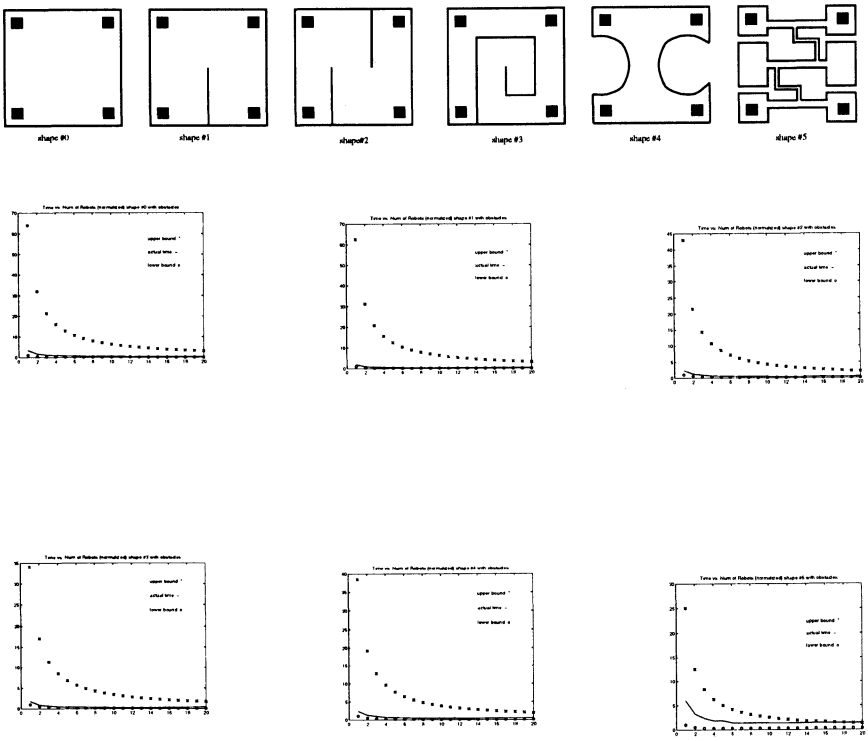

**Figure 16.8** Various shapes with obstacles, tested in **CLEAN-WITH-OBSTACLES** simulations and the normalized time $\left(\frac{t_k}{a}\right)$ vs. number of robots ($k$) for each shape, together with the lower and upper bounds according to Theorem 2

# REFERENCES

[1] B. Bollobas, *Graph Theory - an Introductory Course,* Springer-Verlag, 1990.

[2] G. Beni, J. Wang, "Theoretical problems for the realization of distributed robotic systems," *Proc. of the 1991 IEEE Internl. Conference on Robotics and Automation*, pp. 1914-1920, Sacramento, California, April 1991.

[3] V. Braitenberg, *Vehicles,* MIT Press 1984.

[4] R.A. Brooks, "Elephants Don't Play Chess," in *Designing Autonomous Agents,* P. Maes (Ed.), pp. 3-15, MIT Press/Elsevier, 1990.

[5] R.A. Brooks, "Intelligence without representation," *Artificial Intelligence,* 47 (1991) 139-159, Elsevier.

[6] A.M. Bruckstein, "Why the Ant Trails Look So Straight and Nice ," *The Mathematical Intelligencer,* vol. 15, No. 2, pp. 59-62, 1993.

[7] A.M. Bruckstein, C.L. Mallows and I.A. Wagner, "Probabilistic Pursuits on the Integer Grid," *Technical report CIS-9411, Center for Intelligent Systems,* Technion, Haifa, September 1994. Submitted to *SIAM Review.*

[8] M.P. Do-Carmo, *Differential Geometry of Curves and Surfaces,* Prentice-Hall, New-Jersey, 1976.

[9] M. Held, *On the Computational Geometry of Pocket Machining,* Lecture Notes in Computer Science,Springer-Verlag 1991.

[10] D. Henrich, "Space-efficient region filling in raster graphics," *The Visual Computer (1994)*, 10:205-215, Springer-Verlag 1994.

[11] S. Hedberg, "Robots Cleaning Up Hazardous Waste," *AI Expert* , May 1995, pp. 20-24. Springer-Verlag 1994.

[12] A. Itai, C.H. Papadimitriou, J.L. Szwarefiter, "Hamilton Paths in Grid Graphs," *SIAM J. on Computing (1982)*, 11:676-686

[13] *Instantiating Real-World Agents,* Papers from the AAAI 1993 Fall Symposium, Tech.Rep. FS-93-03, AAAI Press, Menlo Park, California.

[14] W. Neubauer, "Locomotion with Articulated Legs in Pipes or Ducts," *Robotics and Autonomous Systems*, 11:163-169. Elseveier 1993.

[15] S. Levy, *Artificial Life - The Quest for a New Creation,* Penguin Books, 1992.

[16] R.A. Russell, "Mobile robot guidance using a short-lived heat trail," *Robotica* , Vol. 11, 1993, pp. 427-431.

[17] Y. Shoham, M. Tennenholtz, "On Social Laws for Artificial Agent Societies: Off Line Design," to appear at *AI-Journal*, 1995.

[18] H.A. Simon, *The Sciences of the Artificial,* 2'nd ed., MIT Press, 1981.

[19] S. Sen, M. Sekaran, J. Hale, "Learning to Coordinate Without Sharing Information," *Proceedings of AAAI-94*, pp. 426-431.

[20] L. Steels, "Cooperation Between Distributed Agents Through Self-Organization," *Decentralized A.I. - Proc. of the 1'st European Workshop on Modeling Autonomous Agents in Multi-Agent World,* Y. DeMazeau, J.P. Muller (Eds.), pp. 175-196, Elsevier, 1990.

[21] I. A. Wagner and A. M. Bruckstein, "Row Straightening via Local Interactions," *Technical report CIS-9406, Center for Intelligent Systems,* Technion, Haifa, May 1994. Submitted to *SIAM J. on Matrix Analysis.*