

Cooperative Covering by Ant-Robots using Evaporating Traces

Israel A. Wagner *; Michael Lindenbaum † and Alfred M. Bruckstein ‡
Department of Computer Science
Technion City, Haifa 32000, Israel.

July 26, 1996

Abstract

Some insects are known to use chemicals called pheromones for various communication and coordination tasks. In this paper we investigate the ability of a group of robots, that communicate by leaving traces, to perform the task of cleaning the floor of an un-mapped building, or any task that requires the traversal of an unknown network. More specifically, we consider robots which leave chemical odor traces that evaporate with time, and are able to evaluate the strength of smell at every point they reach, with some measurement error. Our abstract model is a decentralized multi-agent adaptive system with a shared memory, moving on a graph whose vertices are the floor-tiles. We describe three methods of cooperatively covering a graph, using smell traces that gradually vanish with time, and show that they all result in eventual task completion, two of them in a time polynomial in the number of tiles. As opposed to existing traversal methods (e.g. DFS), our algorithms are adaptive: they will complete the traversal of the graph even if some of the agents die or the graph changes (edges/vertices added or deleted) during the execution, as long as the graph stays connected. Another advantage of our agent interaction processes is the ability of agents to use noisy information at the cost of longer cover time.

*IBM Haifa Research Laboratory, Matam, Haifa 31905, Israel. *e-mail: wagner@haifasc3.vnet.ibm.com.*

†*e-mail: mic@cs.technion.ac.il.*

‡currently on sabbatical at Bell Laboratories, Murray-Hill, NJ 07974, USA. *e-mail: freddy@cs.technion.ac.il*

1 Introduction

One of the basic theoretical (and practical) problems in multiagent systems is how to design adaptive rules of behavior for the individual, that will lead to the required colony behavior while reducing cost in terms of communication overhead and hardware complexity. We shall consider a cleaning task in which the floor of a building has to be cleaned by a group of autonomous robots which do not have a prior knowledge of the building's floorplan. We model the floor of the building as being composed of small tiles, all of the same size, and assume that a tile can be cleaned in one unit of time. To help their navigation, the robots are allowed to leave traces along their walk, e.g. by means of odor, heat, or color trails. We assume that the intensity of traces decreases with time, therefore by comparing the trace levels at two neighbor tiles, the robot can deduce which tile was visited more recently. The cleanness of the path can also serve as a trace, if the dust is slowly falling back on the floor and hence enables a chronological comparison between tiles. See Figure 1 for an example. The topology of the building may change while the robots work; e.g. people or furniture may move and doors may open or close, hence a preliminary phase of floorplan mapping will not be of much help here. Such a problem is critical in case of a damage to a nuclear reactor, where robots are the only creatures that can survive the radiation and move around to clean hazardous waste [29]. Central control is usually not possible since the strong radioactive radiation avoids almost any possibility of wireless communication. Similarly, one might consider a surveillance task in which robotic guards have to visit the rooms and corridors of a dynamic art gallery, and to guarantee that each and every room and corridor is visited frequently enough.

In this paper we present systematic methods for local, cue-based operation of a group of robots that solves the above problems. As a (simplifying) mathematical model, we use graph traversal, inspired by ant foraging, based on the assumption that the world is divided into vertices (tiles) and edges (tile-separating lines), and an ant leaves a constant amount of pheromone at each point it visits. These traces are later used by the ant and its fellows as a memory of the latest time this point has been visited so far. We shall describe three search algorithms, prove their convergence, bound their worst-case time complexity, and test them empirically.

As far as we know, this work presents the first performance analysis of a model that considers the *cooperative* potential of trace-oriented behavior in terms of time-complexity, under noisy circumstances.

Related Work. (a) **Robotics** An interesting problem of cleaning and maintenance of a system of pipes by an autonomous robot is discussed in [39]. The importance of cleaning hazardous waste by robots is described in [29]. Cleaning and other household robotics applications are discussed in [31]. Another critical application of the covering problem is demining, i.e. the removal of mines from old mine-fields. A centralized multi-robot system for the exploration and cleaning of mines is presented in [40]. Odor marking as an assistance in robot covering and navigation has been presented in [41], where the structure of smell sensors is described. In an earlier work [42], heat trails are used as traces to help robot trail-following. Significant effort has been invested in design, simulation and implementation of multi-agent systems (e.g. [46], [8], [45]). Unfortunately the geometrical theory of such multi-agent systems is far from being satisfactory, as has been pointed out in [5] and many other papers. In [49] an initial step is done towards developing an analytical approach to a cooperative cleaning method where the dirt on the floor is used as a marking. (b) **Computing** Graph search is an old problem; several methods exist for deterministic (e.g. [48], [24]) random (e.g. [2], [11], [13]) and semi-random ([25]) covering, but a lot more needs to be done in order to make the theory useful in the context of robotic covering problems. A step towards a trace-oriented theory of search was done in [10] and [12], where *pebbles* are used to assist

the search. Pebbles are tokens that can be placed on the floor and later be removed. In [12] it is shown that a finite automaton with two Pebbles can search all mazes, but no timing analysis is done. In a sense, our work is a generalization of this work, since one may use “diminishing pebbles” or “deflating tokens” as a model of odor markings. We consider our algorithms to be a reasonable trade-off between the rigid, highly sensitive DFS on one hand, and the absolutely adaptive (but very time-consuming) random walk, on the other hand. Note that the problem of finding the shortest traversal of a graph, is NP-complete even if the graph is completely known, its vertices are grid points and its edges are grid-lines [30]. **(c) Biology** Some aspects of the behavior of continuous smell-oriented swarms were investigated in [37], where a differential equation model is used to investigate the stability properties and the patterns generated in the process. On the biological side, several models have been suggested for the social behavior of ant-colonies, e.g. correlated random-walk in [1]. In a different context, models were investigated for chemosensitive cells (like bacteria or leukocytes) in a random walk that is biased by the concentration of chemicals (e.g. [3]). The foraging and trail-following behaviors have been thoroughly investigated in several works; see e.g. [26] and [21].

The rest of the paper is organized as follows: in the next section we state the problem formally, and state the main results. Section 3 presents the first algorithm, **ANT-WALK-1** with its analysis. Section 4 is devoted to an improved algorithm, **ANT-WALK-2**, and section 5 - to a vertex oriented method of search. Open questions and further research directions are the topic of the discussion in Section 6. In the appendix we discuss an interesting application of trace-oriented walks for creating and maintaining a spanning tree in a dynamic network.

2 Problem definition and summary of results

The task considered here is the traversal of a region by a group of robots which can carry out useful tasks such as cleaning or guarding this region. We are interested in the capabilities of robots working in a distributed unsupervised mode, i.e. the robots determine their motion themselves from local cues and do not rely on external guidance. The aim may be either to cover this region as fast as possible or to move so that all parts of the region will be covered as “uniformly” as possible. This task gets more complicated if there is more than one robot, but, as far as we know, was not considered analytically even in the context of a single robot.

Our approach is computational and consider this task in a generic, abstract context. It uses a graph $G = (V, E)$ to describe the parts of the region to be covered and the connections between them. In this graph every vertex represents an atomic region (“tile”) and every arc represents a neighborhood relation between two such tiles. Our assumption is that the robots are small enough such that several robots can occupy the same tile simultaneously. The task is to cover the graph “as fast as possible”. To achieve this task we specify several strategies. Every such strategy is a definition of a local behavior rule followed by the robot. A strategy is considered better if it covers the region faster, and the efficiency of the different strategies is evaluated by measuring the time from the start until the last yet unvisited tile is reached.

The difference between the strategies we propose is in the assumption we make regarding the amount of memory available to every robot, and regarding the “trace leaving” behavior. Specifically we shall consider either robots which do not have any memory or robots which are equipped with some small memory enabling them to backtrack. We also consider robots which leave traces inside a tile (i.e. on vertices) vs. robots which leave their traces on the passage between different tiles (i.e. on edges).

In analyzing the strategies we shall look for several behavior characterizations: the first obvious

one is that the robots indeed achieve their task and reach every part of the region. We show that this property, denoted *convergence*, indeed holds for all the proposed strategies even though these strategies are extremely simple, and the region is unknown, may be arbitrarily complicated and may even change during the operation. Moreover, the covering processes converge even if some of the robots cease to work before completion; in such a case, the remaining ones will eventually complete the mission.

A second important characterization is the *time* required to cover all the region. We provide here rigorously derived bounds which quantify the time required to cover the region in the worst case. In particular we show that two of the local rules yield a cover of the whole region in time polynomial in the number of tiles. A particular adversary example shows that the quadratic time predicted by this bound is indeed required in some situations. We also simulate the local-rule-driven behavior over several particular scenes and show that in these cases the global performance is actually better.

A third interesting characterization is the *speedup* achieved by using more than a single robot. Both the rigorous bounds and the simulations address this issue and show that a substantial reduction in cover-time is achieved by using more robots. This, however, is only true up to a limit where additional robots are of no help.

The first local rule, denoted **ANT-WALK-1**, requires no individual memory on each machine. The only (shared) memory assumed comprises of the smell traces that are being laid on the edges of the graph G . Denoting by t_k the time needed to cover all edges of the graph by k agents that follow this rule, we prove the following upper bound on t_k :

Theorem 1 :

For **ANT-WALK-1**

$$t_k \leq n\Delta \left(\rho(G) + \frac{(1 + \alpha)n}{k} \right)$$

where Δ is the maximum vertex degree in G , $n = |V(G)|$, α is related to the measurement noise ($\alpha = 0$ implies that the trace intensity sensors are perfectly precise), and $\rho(G)$ is the cut-resistance of G , defined in the sequel and obeying $\rho(G) \leq \frac{n-1}{\lambda(G)}$ with $\lambda(G)$ being the edge-connectivity of G .

The second rule of motion, **ANT-WALK-2**, is a generalization of the common Depth-First-Search method. It relies on a limited amount of memory in each agent, and the ability to control its trace-laying mechanism. In reward, we get the following improved upper bound on the performance:

Theorem 2:

For **ANT-WALK-2**

$$t_k \leq (n\Delta/2) \left\lceil \frac{(1 + \alpha)}{k} \right\rceil$$

where all notations are as before.

The Third method, **VERTEX-ANT-WALK** is similar to the first one, but assumes that smell traces are laid on the vertices rather than the edges. For this method we show that

Theorem 3:

For **VERTEX-ANT-WALK**

$$t_k \leq \frac{n\Delta^d}{k}$$

where d is the diameter of G .

Although the upper bound for **VERTEX-ANT-WALK** is quite high, our simulations show that its performance is actually in the same order of magnitude of the other two algorithms (See Figures 9,10). Another interesting property of the third algorithm is that the paths taken by the robots sometimes converge to cycle-covers of the graph of tiles; see Figures 15-17.

3 ANT-WALK-1: Covering without individual memory

We consider a graph $G = (V, E)$ as a model for the floor to be cleaned. Every vertex in this graph corresponds to a tile on the floor, and every edge - to the boundary between two neighbor tiles. Also, we assign for each edge (u, v) two “smell labels” : $s(u, v)$ which is the time of the most recent traversal of the edge in the u -to- v direction, and similarly $s(v, u)$ for the other direction. All s -labels are initially reset to 0. See Figure 2 for an example of a floor and its graph model. We assume that each time an edge (u, v) is traversed, it is marked by a fresh trace of odor, $s(u, v)$, that overrides the previous trace on this edge in the (u, v) direction; $s(v, u)$ remains unchanged.

We want our group of agents (e.g. ants, robots) to traverse all the edges of the graph without carrying any internal memory. The only traces that are allowed are in the environment - these are the times of most recent visits to each edge in each direction, coded by the trace left there by an agent. In this rule of motion, an ant visiting vertex $u \in V(G)$ checks the labels on all edges emanating from u , in the direction from u outside. Then it goes to an edge that has the smallest trace on it, that is - the edge that was not visited for the longest time. In the course of traversing the edge, the ant leaves there a constant amount of pheromone. The amount of smell on an edge (u, v) , denoted $s(u, v)$, decreases slowly with time, so by “smelling” two edges one can say which one of them was visited before the other. (This ability may be limited by a sensing error α - in the presence of which, one can only distinguish between traces that differ by more than α “smell units”. More on this in section 3.2 in the sequel). Hence, leaving a smell trace on (u, v) at time t is similar to writing the t , the time of traversing this edge. For sake of simplicity, we’ll denote traces by the time they were left. In this discrete setting we assume that if an ant is located at a node $u \in V(G)$ it can move along any of the edges emanating from u to one of its neighbors. The set of u ’s neighbors is denoted by $N(u)$. Formally, the first and simplest local rule of motion to be considered is:

```
Rule ANT-WALK-1(u vertex;
A)  $t := t + 1$ ;
B) find an edge  $(u, v)$  emanating from  $u$  such that
     $s(u, v) = \min_{w \in N(u)} \{s(u, w)\}$ ;
    (if there is more than one such neighbor - make some heuristic decision)
    /* while moving from  $u$  to  $v$ , drop some pheromone along  $(u, v)$  */
C) go to  $v$ , and in the process set  $s(u, v) := t$ ;
end ANT-WALK-1.
```

Note that more than one agent may occupy a tile at the same time; however the agents are exiting the vertex in slightly different times. The order of exiting the tile can be determined by some priority between the agents, which may be determined either by a unique hard-coded ID, a random phase on the clock, or from geometrical considerations (e.g. the agent coming from the north is the first to select an edge, the one coming from the east is the next and so on). The actual way to implement it may be to cause each robot to have a slightly different phase on his clock, or

even use a random phase, which should avoid collisions with high probability. See Figure 2 for an illustration of the algorithm.

3.1 Analysis of the ANT-WALK-1 algorithm

In this subsection we shall prove an upper bound on the time complexity of the ANT-WALK-1 algorithm, assuming noiseless conditions, i.e. $\alpha = 0$. Noisy sensing will be considered in the next subsection. The basic idea in our analysis of the cover time is a proof that the number of passages along one edge cannot differ too much from the number of passages along any other edge in the graph. First, consider the sequence P_u , which is the sequence of vertices visited just after u by all robots, ordered chronologically; that is - each time a robot leaves u and goes to, say, v - then v is added to the sequence. $|P_u|$ will denote the length of this sequence, i.e. the number of times node u has been left so far. Due to the rule of motion defined above, we can show that

Lemma 1 *For each vertex $u \in V$, P_u is a periodic sequence with period d_u , where d_u is the degree of u in G .*

Proof : According to the algorithm, a robot located at u should choose an outgoing edge (u, v) with *smallest* value of $s(u, v)$. Then, in course of traversing the selected edge, the robot sets the label $s(u, v)$ to the current clock value (step C), and, by doing so, makes $s(u, v)$ to be the *largest* among u 's neighbors. So the edge (u, v) now has the lowest priority. In other words, one may define a queue between the edges emanating from u . This queue is prioritized by the value of $s(.,.)$ on each edge, and every time u is visited, the edge on the head of the queue is selected and then this edge is moved back to the tail of the queue. Hence, the edges going out of u are visited in some cyclic order, and this order, once set, is never changed. We conclude that once (u, v) has been visited, it will be visited again after exactly d_u other visits to u . This implies that P_u is periodic with period d_u . \square

From now on we shall not use “smell” or “traces” anymore; rather, we shall only rely on the fact that the neighbors of each edge are always visited in the same cyclic order, no matter what this order is. Hence our results apply as well to any other local decision method that guarantees the cyclic-order property.

Let $f(u, v)$ be the *flow* along edge $(u, v) \in E$, defined as the number of times any robot went over this edge in the direction from u to v ¹. Using P_u to denote the sequence of post- u -visits (as defined above, before Lemma 1), the periodicity proved in Lemma 1 implies that for any neighbor v of u

$$\left\lfloor \frac{|P_u|}{d_u} \right\rfloor \leq f(u, v) \leq \left\lceil \frac{|P_u|}{d_u} \right\rceil, \quad (1)$$

Therefore the ANT-WALK-1 algorithm guarantees that two edges emanating from the same vertex will not differ too much in their respective number of visits, or in other words: the flow from a vertex is fairly distributed among its neighbors.

Lemma 2 *If v and w are both neighbors of u , then, at all times during execution of ANT-WALK-1,*

$$|f(u, v) - f(u, w)| \leq 1. \quad (2)$$

¹Please note that both P_u and $f(u, v)$ are time-dependent; we, however, omit the time parameter in order to simplify our notations.

Proof: implied by Equation 1.

A similar fairness among the edges *entering* a vertex does not necessarily hold.

Remark: The proof relies on the assumption, stated earlier, that several robots may reside in the same tile. Furthermore, during each cycle (i.e. between time t and time $t + 1$), the robots move sequentially and every robot may plan its move based on the traces left by all previous moving robots including those left by robots that move in the same cycle (before its move). As was mentioned above, such a behavior can be achieved with high probability by assigning a random phase to the clock of each robot or, alternatively may be guaranteed by setting hardwired priorities to the robots.

We shall now bound the total flow through any cut in G . Let $S \subset V$ be a (real) subset of the vertex-set of G , and write \bar{S} for $V \setminus S$. We denote by $(S : \bar{S})$ the set of edges having a source in S and a destination in \bar{S} . The flow through the cut is then defined to be

$$f(S : \bar{S}) \triangleq \sum_{x \in S, y \in \bar{S}} f(x, y).$$

Lemma 3 *At all times, and for all cuts $(S : \bar{S})$ in G ,*

$$\left| f(S : \bar{S}) - f(\bar{S} : S) \right| \leq k$$

where k is the number of agents travelling in G .

Proof: For each agent a_i , let us denote by $f_i(x, y)$ the number of times it traversed the edge (x, y) in this direction. Considering a cut $(S : \bar{S})$, it clearly holds that the number of times an agent has crossed the cut in one direction cannot differ by more than 1 from the number of crossings, by the same agent, in the opposite direction:

$$\left| \sum_{(x,y) \in S} (f_i(x, y) - f_i(y, x)) \right| \leq 1.$$

Summing over all the k agents, we get the Lemma. □

Combining the fairness of flow (Lemma 2) and its boundedness (Lemma 3), shows that the intensity of flow cannot differ too much between vertices in G .

Let $g(x)$ denote the maximum flow along an edge emanating from a vertex $x \in V$

$$g(x) = \max_{y \in N(x)} \{f(x, y)\}.$$

The flow $f(x, y)$ associated with every edge (x, y) emanating from x satisfies $g(x) - 1 \leq f(x, y) \leq g(x)$ (Lemma 2). Let $S[1, n]$ be the set of all vertices $\{x_1, \dots, x_n\}$ ordered in increasing order of $g(\cdot)$, i.e.

$$g(x_1) \leq g(x_2) \leq \dots \leq g(x_n).$$

Let $S[i, j]$ be the subset $\{x_i, x_{i+1}, \dots, x_j\}$, and consider the cut $C(i, j)$ between $S[1, i]$ and $S[j, n]$,

$$C(i, j) = \{(x_p, x_q) \mid p \leq i, q \geq j\}.$$

In the following lemma we show that the function $g(\cdot)$ has a discrete “smoothness” property:

Lemma 4 *At all times and for all $1 \leq i \leq n$*

$$|g(x_i) - g(x_{i+1})| \leq 1 + \frac{k}{|C(i, i+1)|}.$$

Proof: If $g(x_i) = g(x_{i+1})$ then our Lemma is clearly true. Otherwise, $g(x_i) < g(x_{i+1})$ and for each edge (x_p, x_q) in $C(i, i+1)$ (i.e. $p \leq i, q \geq i+1$) we have that $f(x_q, x_p) - f(x_p, x_q) + 1 \geq g(x_{i+1}) - g(x_i)$. The same is true for all the edges in this cut; hence the net flow across the cut $C(i, i+1)$ is lower-bounded as follows:

$$|f(S[1, i] : S[i+1, n]) - f(S[i+1, n] : S[1, i])| \geq |C(i, i+1)| \cdot (g(x_{i+1}) - g(x_i) - 1).$$

But this net flow is also bounded above by k (Lemma 3), so we get that

$$|C(i, i+1)| \cdot (g(x_{i+1}) - g(x_i) - 1) \leq k$$

and hence

$$g(x_{i+1}) - g(x_i) \leq 1 + \frac{k}{|C(i, i+1)|}$$

which yields the lemma. \square

It follows that the difference between the g -values of any two nodes in G cannot become too large:

Corollary 1

$$\forall x, y \in V : |g(x) - g(y)| \leq n - 1 + \sum_{i=1}^{n-1} \frac{k}{|C(i, i+1)|}$$

where n is the number of vertices in G .

Also, since $|C(i, i+1)|$ is bounded below by $\lambda(G)$, the edge connectivity of G ², it can easily be seen that

Corollary 2

$$\forall x, y \in V : |g(x) - g(y)| \leq (n - 1) \left(1 + \frac{k}{\lambda(G)} \right).$$

Using the edge connectivity λ has a disadvantage: its value for G is strongly biased by extremal areas of the graph, e.g. one narrow corridor may impose a low λ value on an otherwise dense graph. Rather, we suggest a more realistic measure of the “resistance” of G to the motion of trace-oriented agents in it. For this purpose we define the *cut-resistance* of a graph as follows:

Definition 1 *Assume that $\{x_1, \dots, x_n\}$ is the set of vertices in a graph G . For a given permutation $\sigma \in S_n$ we define the cut-resistance induced by this specific permutation to be*

$$\rho_\sigma \triangleq \sum_{i=1}^{n-1} \frac{1}{|C(\sigma_i, \sigma_{i+1})|}$$

and the cut-resistance of G is defined as the maximum of ρ_σ over all possible permutations in S_n :

$$\rho(G) \triangleq \max_{\sigma \in S_n} \{\rho_\sigma(G)\}.$$

²The edge-connectivity of a connected graph G is defined [6] as the minimum number of edges that need to be removed in order to disconnect the graph, i.e. if G has edge connectivity $\lambda(G)$, there is no set of $\lambda(G) - 1$ edges the deletion of which will disconnect G .

The following observations on $\rho(G)$ can easily be verified:

- if G is a path along n vertices, then $\rho(G) = n - 1$.
- if G is a circle with n vertices, then $\rho(G) = (n - 1)/2$.
- if G is the complete graph (K_n), then

$$\rho(G) = \frac{1}{(n-1)} + \frac{1}{2(n-2)} + \frac{1}{3(n-3)} + \dots + \frac{1}{(n-1)} < 1.$$

- if G is an $m \times m$ grid-graph then $\rho(G) = O(m) = O(\sqrt{n})$.
- for every graph, $\rho(G) < (n - 1)/\lambda(G)$, since any cut has at least $\lambda(G)$ edges in it.

Intuitively: the denser the graph, the lower its cut-resistance.

The following corollary results from Lemma 4

Corollary 3

$$\forall x, y \in V : |g(x) - g(y)| \leq n - 1 + k\rho(G).$$

Recall that $g(u)$ is the maximum flow along an edge emanating from u , and that this amount (according to Lemma 2) is at most 1 unit more than the minimum flow along an edge emanating from u . Hence we get

Corollary 4

$$\forall (u, v), (x, y) \in E : |f(x, y) - f(u, v)| \leq 1 + k\rho(G) + (n - 1) = k\rho(G) + n.$$

Remark: Another function that could come to mind here is the *edge-expansion factor* of the graph, defined as

$$\min_{S \subset V, |S| \leq n/2} \left\{ \frac{|(S : \bar{S})|}{|S|} \right\}.$$

but it should be noted that $\rho(G)$ is better suited for our purposes, since it represents the conductivity of the whole graph, not only the extremal (i.e. worst) regions in it. For example, consider a graph which constitutes of two complete subgraphs G_1, G_2 each isomorphic to $K_{n/2}$, with one edge connecting them. The edge-expansion factor of G will be as small as $2/n$, while its cut-resistance will not be much different from that of the complete graph K_n , since almost any cut has at least $n/2$ edges in it.

Let $t_k(G)$ denote the time needed to cover the edges of a connected graph G by k agents that follow the ANT-WALK-1 algorithm. The following theorem establishes an upper bound on this time.

Theorem 1

$$t_k \leq n\Delta \left(\frac{n}{k} + \rho(G) \right)$$

where Δ is the maximum vertex degree in G , $n = |V(G)|$ and $\rho(G)$ is the cut-resistance of G .

Proof: Once started, our agents never rest; they traverse an edge in every clock cycle. Even when two (or more) agents occupy the same tile, they can both use the next cycle because we assume that they have different clock-phases. Hence, after t units of time, the total flow in the graph is

$$\sum_{(x,y) \in E} f(x,y) = kt$$

(Recall that $f(x,y)$, the flow along an edge (x,y) , is defined as the number of passages so far along the edge in the x -to- y direction). Assume, in contradiction, that the time t_k specified by the theorem has passed, yet there is an edge (x,y) such that $f(x,y) = 0$. Corollary 4 implies that no edge has a flow larger than $k\rho(G) + n$. Hence, the total amount of flow in G can be bounded from above:

$$\begin{aligned} \text{total flow in } G &= \sum_{(u,v) \in E} f(u,v) \leq \\ &\leq (|E(G)| - 1)(k\rho(G) + n) \leq \\ &\leq (n/2)\Delta(k\rho(G) + n) \end{aligned}$$

(using $|E(G)| \leq n\Delta/2$). Dividing by k we get that the time passed cannot exceed $(n/2)\Delta(\rho(G) + n/k)$, in contradiction with the assumed time. \square

Remark 1: The bound we proved is tight up to a constant, as can be seen from the example in Figure 3 where an ant goes back and forth several times before covering the graph. Such problems can sometimes be cured using a heuristic to resolve ties among edges with equal traces on them. Another solution is to allow backtracking as will be shown in algorithm ANT-WALK-2 that is described in the next section.

Remark 2: The expression in parentheses, $(\frac{n}{k} + \rho(G))$, can be explained intuitively as follows: If the graph is dense, its cut-resistance $\rho(G)$ is small and the term $\frac{n}{k}$ is significant, i.e. - more robots deliver a faster cleaning. On the other hand if G is sparse $\rho(G)$ is large and increasing the number of robots does not make much sense. This intuitive interpretation was also observed in our simulations.

3.2 The effect of noise and sensing-errors on the dynamics of ANT-WALK-1

In reality, sensors and effectors are not perfectly reliable and noise can distort their operation. The following a rather skeptical statement was made in [14]:

...[sensors]... simply do not return clean accurate readings. At best they deliver a fuzzy approximation to what they are apparently measuring, and often they return something completely different.

However, the effect of noise depends on the type of algorithm that is being used in the system. In a multiagent system the noise has another aspect - the individual agent cannot always be sure as for the reliability of the others, hence a good multiagent algorithm should be tolerant to failures occurring in inter-agent communication. Formally, if one agent has a fault probability of p , then an algorithm that relies on the correct behavior of all k agents will have failure probability of $1 - (1 - p)^k$, which tends to 1 as k grows.

We propose that our approach to covering problems is well-suited for noisy sensors/environments, since it does not rely too strongly on any specific data or hardware, but only on relative quantities

which do not suffer so much from the errors in sensor readings. Let α be the amount of noise in the sensor measurement. That is, if the smell along edge (x, y) is equal to $s(x, y)$, then the noisy sensor may tell us anything between $s(x, y) - \frac{\alpha}{2}$ and $s(x, y) + \frac{\alpha}{2}$. Such a level of noise implies that a sensor (or an algorithm that relies on the sensor) cannot distinguish between a trace that was left at some time t and one that was left at time $t + \alpha$. This causes a deviation from the basic rule of behavior which implies that the difference in flow (i.e. number of passages) between two edges emanating from the same vertex may become as large as $1 + \alpha$ (rather than 1 as before). In order to estimate the quantitative effect of such noise on the performance of our ANT-WALK-1 algorithm, we rework our previous results to account the possible deviation.

Lemma 2. α *If v and w are both neighbors of u , and the sensor deviation is at most α , then, at all times during execution of ANT-WALK,*

$$|f(u, v) - f(u, w)| \leq \alpha + 1. \quad (3)$$

This modification is due to the fact that an edge may be traversed up to α times before the noisy sensor can see that this edge is favored over the others emanating from u .

Consequently, Lemma 4 becomes

Lemma 4. α *With sensor deviation at most α , it always holds that for all $1 \leq i < n$*

$$|g(x_i) - g(x_{i+1})| \leq 1 + \alpha + \frac{k}{|C(i, i+1)|}.$$

and

Theorem 1. α

$$t_k \leq n\Delta \left(\frac{(1 + \alpha)n}{k} + \rho(G) \right).$$

where α is a bound on the deviation of the sensors.

Observe that as the noise parameter α grows, there is more sense in using a larger number of robots. Interestingly enough, this observation is similar to the experimental results presented in [18] (in a rather different context), where a crypt-arithmetic puzzle is being solved by several cooperating agents that can exchange hints over a common blackboard. There, it was observed (empirically) that a group of solvers may demonstrate a better cooperation under noisy conditions.

4 ANT-WALK-2: covering with backtracking - DFS extended to multilevel search

As we saw in the previous section, there are cases where ANT-WALK-1 gets into a series of “traps” that cause a useless re-visiting of the same locations. In this section we suggest another algorithm that, in general, works more efficiently, but requires some more complicated hardware. This algorithm is essentially a generalization of the famous Depth-First-Search algorithm. First, let us formulate the common DFS (as in [47],[28],[23]). The basic idea in Depth-First-Search is to try and continue the search to a neighbor that has not yet been visited; if none exists, the search “back-tracks” along the edge used in the first entrance to the current vertex. If no such edge exists (i.e. the current vertex was the first in the search), the search terminates and reports that all edges of G have been visited. To formalize it in our trace-oriented terms, we will need the following two

definitions, that make use of the traces on the edges: for each vertex u , let $t_{in}(u)$ be the time of the first entry to u , i.e.

$$t_{in}(u) \triangleq \min_{v \in N(u), s(v,u) > 0} \{s(v,u)\},$$

and let $t_{out}(u)$ be the time of the first exit from u , i.e.

$$t_{out}(u) \triangleq \min_{v \in N(u), s(u,v) > 0} \{s(u,v)\}.$$

Recall that in the DFS algorithm an edge is never visited twice in the same direction (Lemma 3.1 in [23]), hence the t_{in} and t_{out} values, once set, will never change. We assume that initially all t_{in} and t_{out} values are set to 0. Two observations can be made:

1. If both $t_{in}(u)$ and $T_{out}(u)$ are 0 then u is a “new” vertex, i.e. it has never been visited.
2. If $t_{in}(u) > t_{out}(u) > 0$ (i.e. it was left before it was entered) then u was an initial vertex of the tour.

The common DFS can now be formulated as follows:

```

Rule DFS(u vertex;)
A)  $t := t + 1$ ;
B) if  $\exists v \in N(u)$  s.t.  $s(u, v) < 1$ 
    then
        set  $s(u, v) := t$ ;
        if  $t_{in}(v) < 1$  go to  $v$ ;
                                     /* am I back in the origin ? */
C) if  $t_{in}(u) > t_{out}(u) > 0$  then STOP. (the graph is covered).
                                     /* backtracking - all neighbors are old */
D) find an edge  $(u, v)$  such that  $s(v, u) = t_{in}(u)$ ;
E) set  $s(u, v) := t$ ;
F) go to  $v$ .
end DFS.

```

Note that in order to perform a proper backtracking, one *must* be able to mark the entry edge for each vertex. But if, for some reason, this mark is lost, or the graph is changed, then the desperate searcher is hopeless and will never cover the graph. Hence, the DFS is not suitable for a noisy environment where marks and traces are prone to frequent change or misinterpretation. Another problem in using DFS for multi-robot covering task is how to apply it for several cooperating robots. The reason is that once a robot got back to its starting point, it will (according to DFS) stop there forever, rather than go around and help his hard-working fellows.

In order to overcome the above disadvantages a *multilevel DFS* approach is suggested. In this method, when an agent r is facing a situation in which the search cannot continue (i.e. no backward edge emanates from the current vertex), then a new level of search is started by increasing the value of the **search-level**(r) variable, which is individual for robot r . This variable stores the time when “the new history begins” i.e. any edge/vertex visited before that time is now considered un-visited by robot r , as opposed to the common DFS where all visited vertices are considered “visited” in exactly the same way. Hence, if $t_{in}(u) < \mathbf{search-level}(r)$ for a robot r , then the vertex u is considered “new” by this robot. Initially, all search-level variables are set to 0, and the rule of motion for each agent is:

```

/* multilevel DFS -
   initially all search-level's are set to 1, and all  $s(.,.)$ 's to 0. */
Rule ANT-WALK-2(search-level( $r$ ) integer,  $u$  vertex;)
A)  $t := t + 1$ ;
B) if  $\exists v \in N(u)$  s.t.  $s(u, v) < \mathbf{search-level}(r)$ 
   then
       set  $s(u, v) := t$ ;
       if  $t_{in}(v) < \mathbf{search-level}(r)$  go to  $v$ ;
       /* have I exhausted the current level of search ? */
C) if  $t_{in}(u) > t_{out}(u) \geq \mathbf{search-level}(r)$  then
       set search-level( $r$ ) :=  $t$ ;
       /* backtracking - all neighbors are old */
D) find an edge  $(u, v)$  such that  $s(v, u) = t_{in}(u)$ ;
E) set  $s(u, v) := t$ ;
F) go to  $v$ .
end ANT-WALK-2.

```

Note that Step B is taken if u has a neighbor not visited in the current search level, Step D is a backtracking step, and (in Step C) the **search-level** of agent r is increased if the current level of DFS cannot be continued and a new level of search has to be established. This may happen in one of three cases:

1. The robot got back to the point where the current level of search has started. It should now start a new search by setting a new value of **search-level**, (rather than take a nap) in order to help other agents.
2. A change has occurred in the graph (e.g. an obstacle has been moved, which, in our model, means that an edge/vertex has been added or deleted) that makes it impossible to backtrack as usual.

3. The noise in the sensors makes the robot believe that option 1 above is true.

Note that in this second algorithm each robot needs to remember its search-level, and to make more calculations in each vertex it visits. However, the reward is a better performance as will now be shown.

4.1 Analysis of ANT-WALK-2

First assume that $k = 1$ (one agent) and there is no change in the graph. Then ANT-WALK-2 is just another variation on Depth-First Search, that covers the graph in time at most $2|E|$ (as proved in Lemma 3.1 of [23]), then starts a new tour and so on.

For more than one agent, the work may or may not be distributed between the agents. In the worst case, they will just repeat each other's steps and create (at most) k levels of search before full covering is achieved. In this case there will be no speed-up. (an example for such a miserable case is when all robots are initially placed near one end of a long and narrow corridor). But even if in general the work is not necessarily evenly-distributed between the agents, the increase in number of agents may be useful to reduce the effect of noise. Assume, as before, that the sensors are prone to an error of up to α units, as defined in Section 3.2. Then in the worst case, α levels of search are needed to guarantee a full coverage of the graph. Formally,

Theorem 2 *k agents obeying ANT-WALK-2 and having sensory noise level of at most α units will cover a graph G in time t_k where*

$$t_k \leq (n\Delta/2) \left\lceil \frac{1 + \alpha}{k} \right\rceil.$$

Proof: If $\alpha = 0$ then ANT-WALK-2 cannot be worse than the common DFS which is known to cover the graph in time bounded above by $2|E| \leq n\Delta/2$. If there is noise, i.e. $\alpha > 0$, then, in the worst case, an edge may need to be traversed up to $1 + \alpha$ times per search-level before the robot can "see" that this edge has indeed been visited. Therefore no edge is traversed more than $1 + \alpha$ times before all its neighbors are traversed once. On the other hand, k robots traverse k times more edges than one does, hence their cover time t_k cannot exceed $\left\lceil \frac{1+\alpha}{k} \right\rceil$ times t_1 - the cover time of one noise-less robot. \square

See Figures 6-7 for the evolution of ANT-WALK-2 with varying amount of noise, and Figures 9-13 for a plot of the cover time vs. level of noise, for various noise levels and numbers of robots. Figure 14 shows how a group of ants obeying the ANT-WALK-2 rule can overcome a change in the topology of the region to be covered.

5 VERTEX-ANT-WALK - A Vertex-oriented search

In certain scenario there is no need to cover all *passages* between the tiles and the previous algorithms, which aim to cover all edges require too much. Therefore, one can suggest another algorithm which makes its choice between all accessible *tiles* and uses a trace left on the *tile* itself as a basis for its decision. The following VERTEX ANT WALK algorithm uses this principle. The trace $s(u)$ left in the vertex $u \in V(G)$, is, again, the time of the visit t . The local decision rule for the ant being in location u is to go to the neighbor with the least t -value. Formally,

Rule VERTEX-ANT-WALK(**u** vertex;)

A) $t := t + 1$;

B) find a vertex v in $N(u)$ such that

$$s(v) = \min_{w \in N(u)} \{s(w)\};$$

(if there is more than one such neighbor - make some heuristic decision)

/* now drop some trace on u */

C) set $s(u) := t$;

D) go to v ;

end VERTEX-ANT-WALK.

The theorem provides a worst case bound which is exponential in the graph diameter. We believe that this bound is very far from tight. The simulations we conducted show that this algorithm performance is close to those associated with the edge based algorithms, although it is sometimes inferior, especially when sensor inaccuracy is high.

A system of agents obeying this rule is guaranteed to cover the vertices, and works quite well in simulations. Its covering time is upper bounded in the following theorem.

Theorem 3 *Following the VERTEX-ANT-WALK rule, a group of k agents will cover the vertex set of a graph G within time t_k , such that*

$$t_k \leq \frac{n\Delta^d}{k}$$

where n is the number of vertices, Δ is the maximum degree and d is the diameter of G .

Proof: (a) If (u, v) is an edge in G then u should be visited at least once every Δ visits to v , since after each visit to v one of its neighbors is visited and hence after Δ visits to v , if u has not been visited so far, all v 's neighbors have s -value greater than $s(u)$, hence u should be visited no later than after the next visit to v . Hence we get

$$f(u) \leq \Delta(f(v) + 1) \tag{4}$$

where $f(x)$ denotes the number of visits to node x so far.

(b) Let us assume that some vertex, say x_1 , has not yet been visited, hence $f(x_1) = 0$. Now consider the farthest vertex from x_1 , say x_q , and a shortest path between them $P = x_1, x_2, \dots, x_q$. Clearly q , the length of the path, is smaller than or equal to d , the diameter of G . Using Equation 4 we get

$$f(x_q) \leq \Delta + \Delta f(x_{q-1}) \leq \Delta + \Delta^2 + \Delta^2 f(x_{q-2}) \leq \dots \leq \Delta^q + f(x_1) \leq \Delta^d + f(x_1). \tag{5}$$

But since $f(x_1) = 0$ we have $f(x_q) \leq \Delta^q$, and the total amount of visits in G is hence bounded above by $\Delta^d(n - 1)$.

(c) Assuming that k agents are active on the graph, after t units of time there have been a total of kt visits to vertices in the graph, hence

$$t_k \leq \frac{n\Delta^d}{k}$$

□

We assume that initially all agents are located on distinct vertices, hence when $k \rightarrow \infty$, t_k goes to 0 as the cover is immediate.

The above is only a worst-case upper bound; Actually, simulations show a much better behavior. This gives rise to the hope that a tighter upper bound will eventually be discovered.

The dynamics involved with this vertex-orient ant-walk has the interesting property of having cycle covers of the graph among its fixed points, i.e. once the k agents are in a loop of one (or more, up to k) cycle(s), they continue hopping on these cycles forever. In the case of a single agent, such a cycle is a Hamiltonian cycle; See Figure 15. For a $k > 1$, its a 2-factor (or “cycle cover”) of the graph. See Figures 16 - 17 for some examples. Hence, such a dynamic may serve as a heuristic for finding a Hamiltonian path.

6 Simulations and Experiments

We see the main goal of this paper in developing the necessary theory for proving upper bounds on the covering times of our algorithms. However, experiments are useful for getting an idea on the practically more interesting average behavior of the group of robots. Hence, the three ANT-WALK algorithms were implemented in the *C* programming language on an IBM-Power2 workstation under the AIX operating system. The algorithms were tried on several shapes and agent-numbers, with a subset of the integer lattice as the underlying graph. The numerical results of cover time are presented in Figures 7 - 13. An important property of our algorithms is their ability to overcome noisy conditions, so we simulated each rule on three levels of noise, namely 0,10 and 20 units. Due to the random nature of the noise, it is also important to run the algorithm several times and find the average time of covering. A comparison between one-shot covering times (Figures 7 - 10) and average covering time (Figures 7 - 13) shows that, on the average, more robots bring a faster covering period.

In Figure 9 we show the time of covering vs. the number of agents, with varying amount of noise for the three algorithms. Noise is simulated in the following way: when a noise-level of α is assumed, and the actual smell is s , the sensor reading is interpreted as $s + p\alpha/2$, where p is a random number between -1 and 1 . The plots in Figures 10 and 13 show the same information but ordered by noise-level; it shows that ANT-WALK-1 and -2 are far better than VERTEX-ANT-WALK in noisy environment, while the latter is better when no noise is present.

Specific examples are shown in Figures 4 - 7. In these figures, the gray level in a tile represents the trace intensity i.e. the time since last visit to this point. Figure 14 shows an example with a blocking obstacle that is removed during execution. It can be seen that the agents detect the change in topology by doing a multi-level DFS, following the ANT-WALK-2 rule. The other algorithms will have no problems with this situation as well.

In Figures 15-17 we show examples of the VERTEX-ANT-WALK dynamics with the cycle-cover that emerges as a fixed point of the process. Note that the process does not necessarily converge to a cycle-cover even if one exists in the graph.

We are now in the process of building an experimental robotic vehicle that is able of laying traces and using them for navigation. This will serve to test our model and algorithms in a “real-life” situation.

7 Discussion

We addressed the problem of exploring an unknown area, with simple robots that can leave and sense traces on the ground, and this is their only way of communication. We have shown that even such simple imitations of ants can cooperate efficiently in their mission of exploration. We proved that a group of k such ants, obeying either the ANT-WALK-1 or ANT-WALK-2 rule, covers a graph in polynomially-bounded time. For a third algorithm, VERTEX-ANT-WALK, we showed an exponential upper bound and presented a fascinating property of its limit behavior, namely that cycle-covers of the graph are among the limit cycles of this process. Another application to the trace-oriented cover methods is the maintenance of spanning tree that can recover from changes in the graph.

An advantage of our algorithms over existing search methods is in their adaptivity to changes in the environment and to noise in the reading of the sensors. This property comes from our usage of *relative* rather than absolute value of traces.

Although we assumed that the explored area is divided into equally-shaped tiles in the form of a grid, this is not at all necessary; actually our algorithms and proofs apply to any set of local missions that are geometrically connected by neighborhood relations. For example, we can associate *rooms* (rather than tiles) with the vertices in the graph, and instruct the robot to clean the full room before leaving it. If we define a room to be, say, an $m \times m$ square array of tiles, then the effort at every vertex is multiplied by m^2 , but the number of vertices is now m^2 times smaller. Recall that the worst case complexity of the first algorithm depends on the square of the number of vertices; this makes the total complexity much smaller.

Our model was inspired by ants behavior, but can be used to design practical multi-agent robotic systems. Some of the potential applications of the smell-oriented navigation model are in cooperative cleaning of a dirty region with obstacles, and in the maintenance of spanning tree in a communication network (see the Appendix).

There are several other issues that may be of interest for a practical implementation of covering algorithms, and should be further investigated by both analysis and simulations:

- **Continuous trace-oriented walk:** our analysis referred to discrete tiles on a grid. A challenging question is whether a continuous version of one (or more) of our algorithms will work in a continuous setting, and how fast can it be. Our simulations on such a setting show very good convergence times, but a rigorous quantitative analysis will probably give the necessary insight into the dynamics.
- **Rate of covering:** the amount of covering per unit of time is not at all constant during execution, as can be seen in Figure 4, where more than 50% of the area is covered within 20% of the total covering time. In some applications (e.g. surveillance) it may be better to choose a “quick and dirty” algorithm that covers a significant part of the area in a short time, rather than one with a shorter time of total covering.
- **Load balancing:** Do all robots invest about the same effort ? We would clearly like to have as fair work distribution as possible.

- **Dependency on the shape:** as shown in our analysis, the upper bound on covering time depends not only on the area of the region but also on its shape, as represented by the cut-resistance parameter ρ (see Theorem 1). Calculating ρ analytically is possible only for simple cases; however our simulations show that as the shape becomes more complicated (and hence ρ grows) it takes a longer time to cover. But this point needs a more detailed quantitative investigation; i.e. taking several shapes (with the same area) and comparing their covering time and ρ -values.

The model we described in this paper is fairly simple but seems to yield interesting results and to pose intriguing challenges for both theoreticians and implementers of robotic systems.

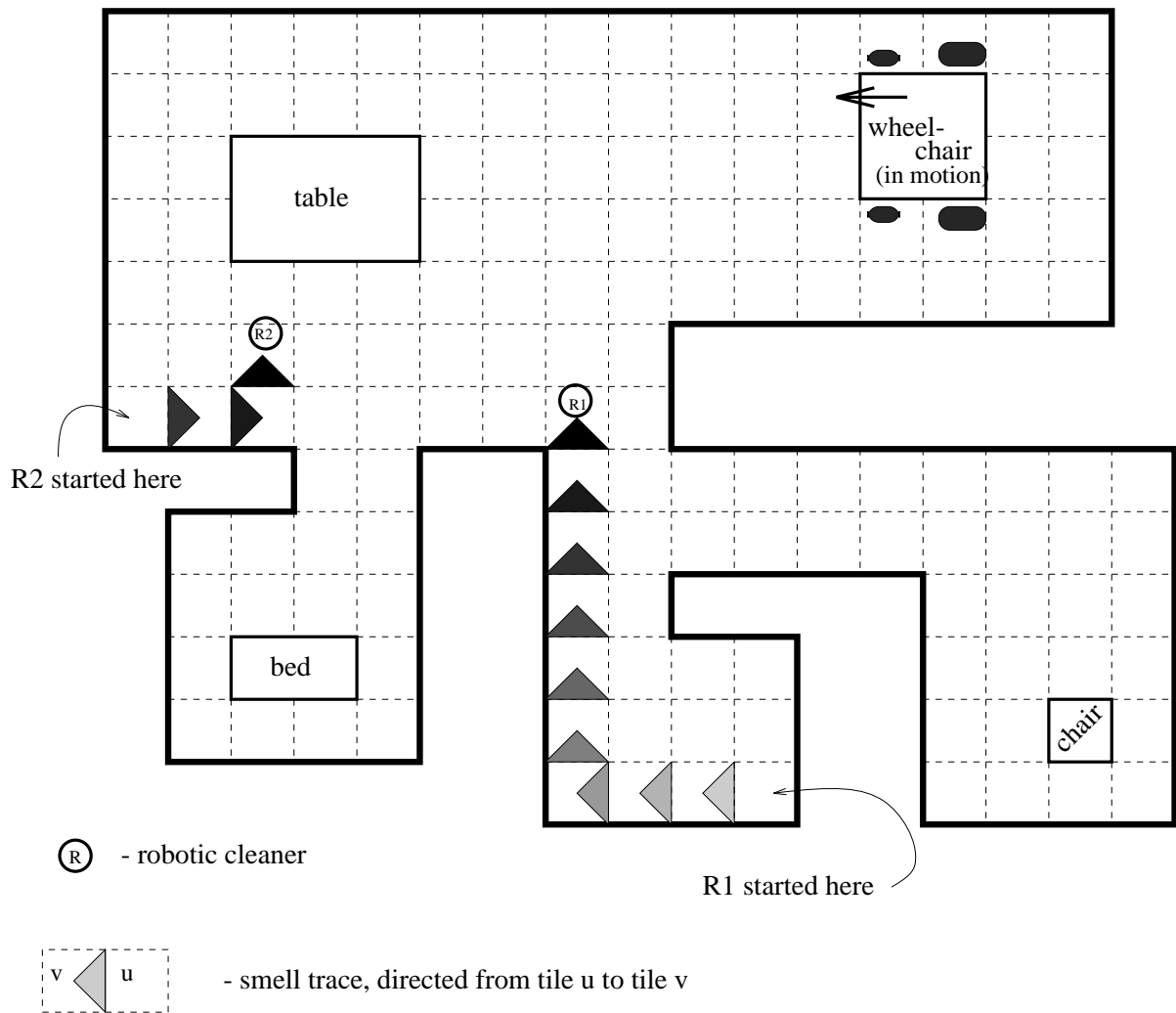


Figure 1: A system of rooms divided into square tiles, with a dynamic obstacle. Two cleaning robots are shown with their directional smell traces. Note that the traces degrade with time.

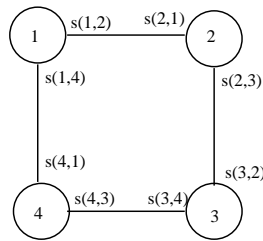
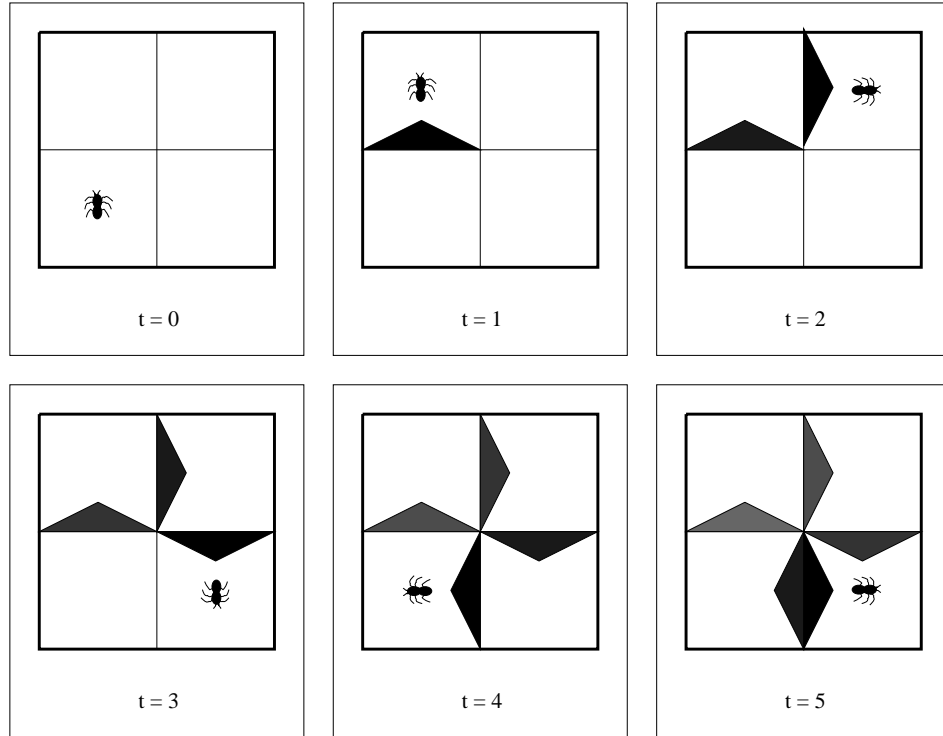


Figure 2: Four cycles of ANT-WALK-1 are needed to traverse a floor with 4 tiles. Also shown is the corresponding directed graph with the smell-labels on its edges. Note that there are two labels on each edge, to designate the trace intensity in each direction of the edge.

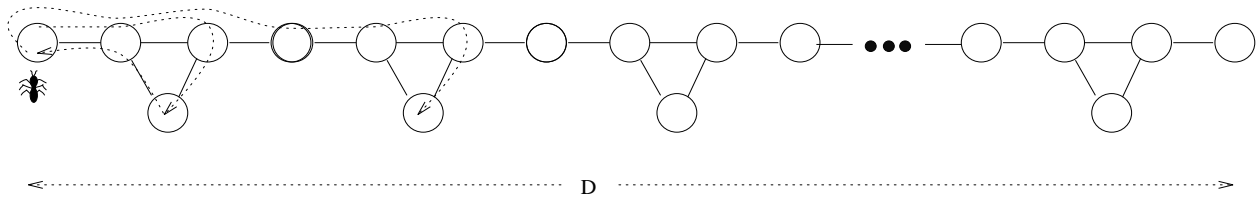
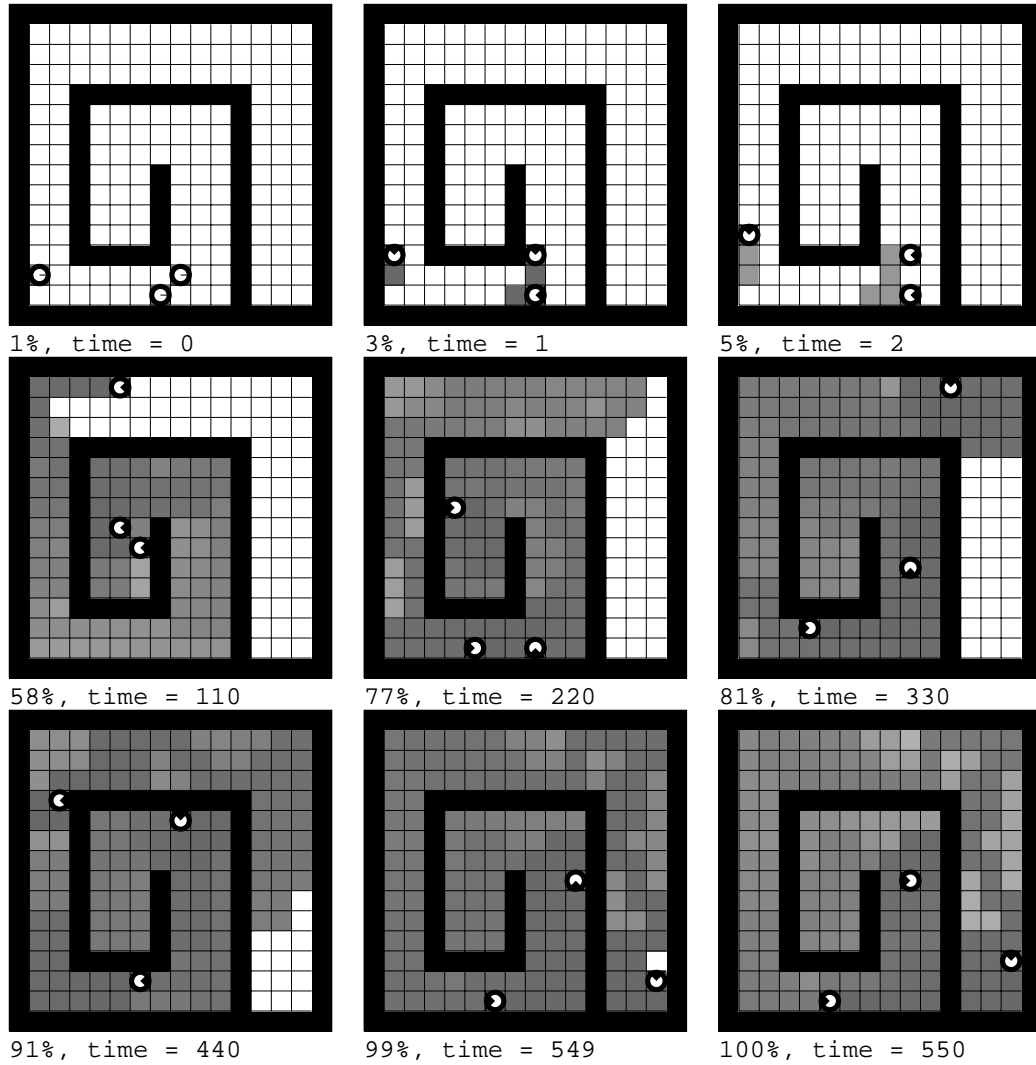


Figure 3: A hard case for the ANT-WALK-1 rule. There are n vertices, and about $1.25n$ edges. The diameter is about $0.8n$, and the time needed to traverse it may be as long as $O(n^2)$. The dotted arrows show the worst case where each triangle of vertices is a “trap” that causes the ant to go back to its starting point.



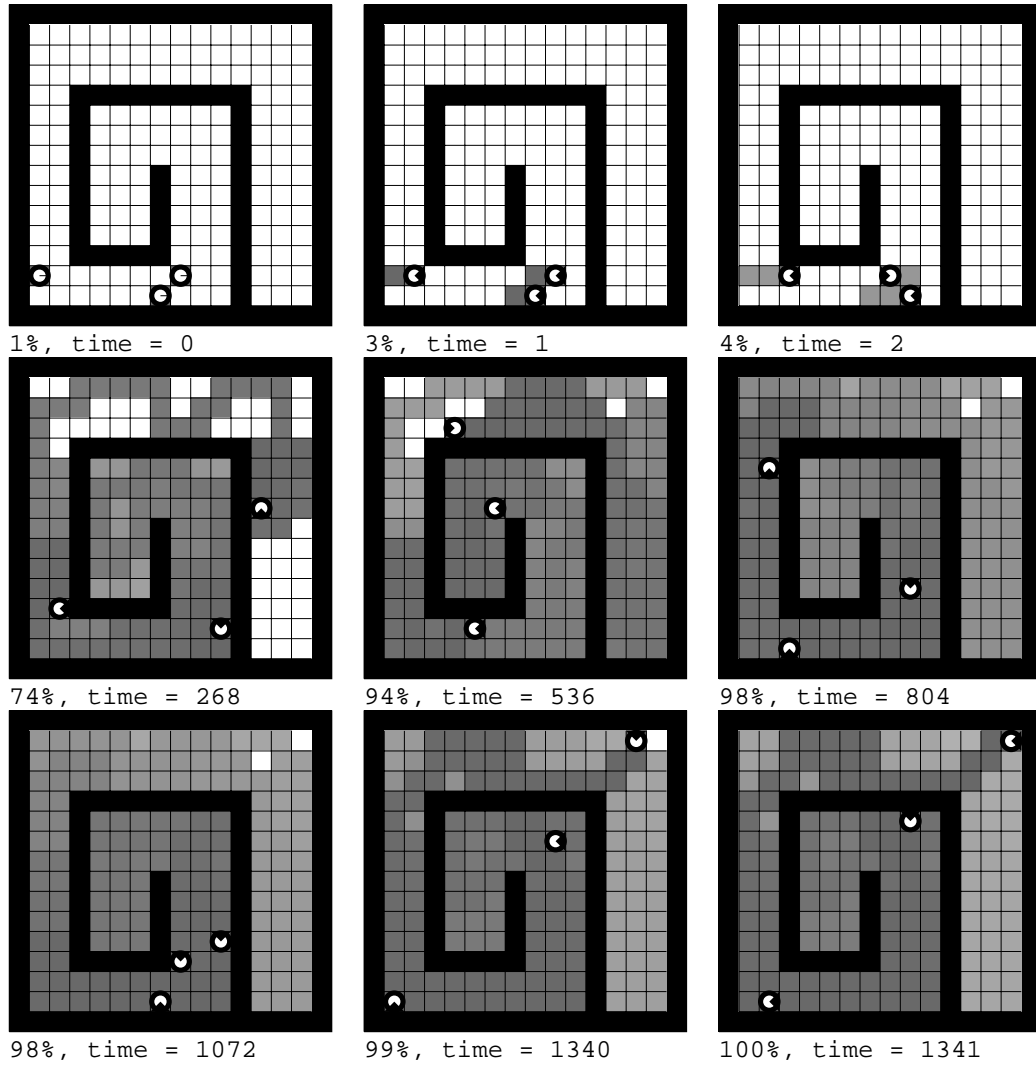
Covering by ANT-WALK-1, shape #3, on a 14 x 14 matrix

Num of Ants = 3; Total area = 161; Cleaned area = 161

Noise level = 0 units; Cover time = 550;

Rule: ANT_WALK_1 (no backtracking);

Figure 4: Simulated evolution of three agents, oriented by the ANT-WALK-1 rule. The gray-level is proportional to the number of visits in each tile. Note that most ($\approx 90\%$) of the area is cleaned within 50% of the time.



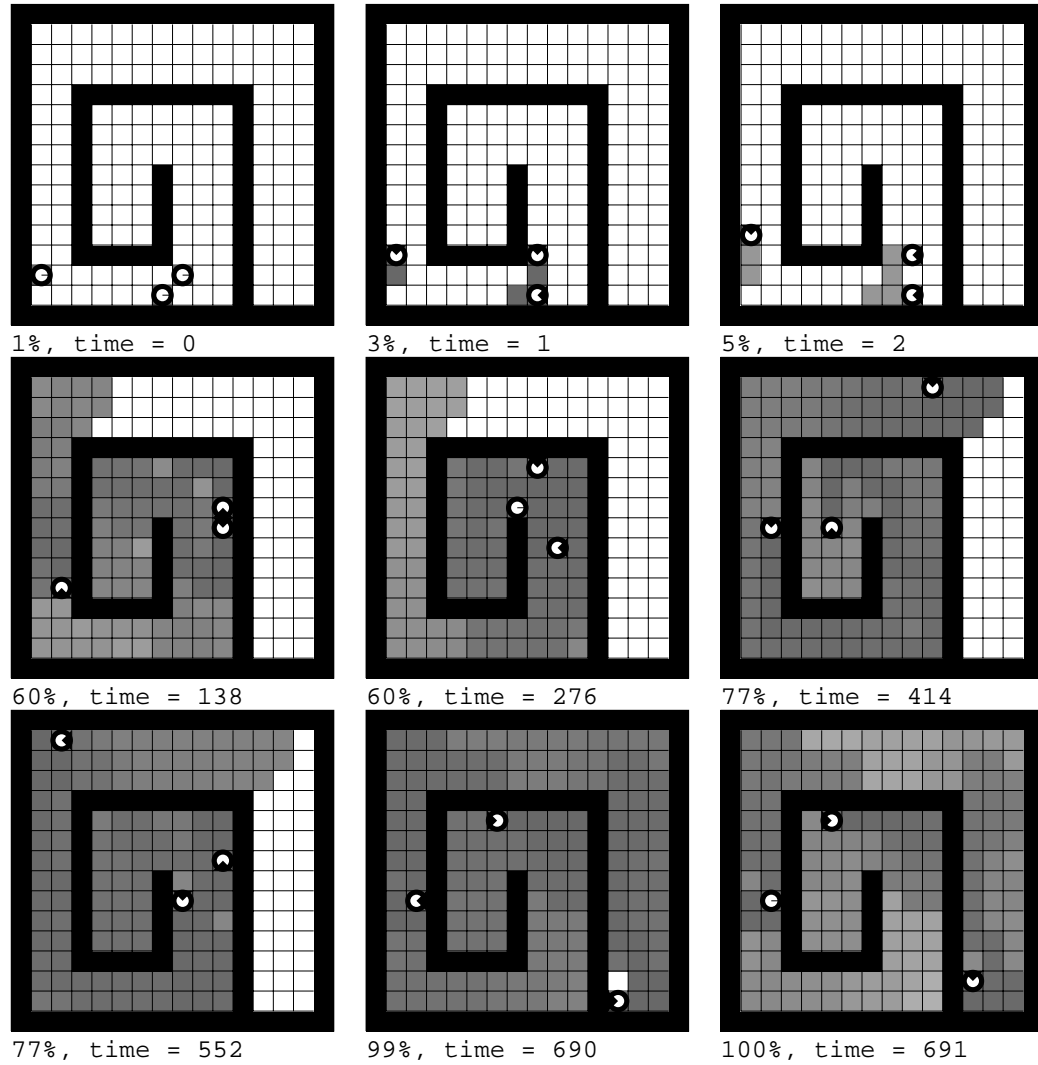
Covering by ANT-WALK-1, shape #3, on a 14 x 14 matrix

Num of Ants = 3; Total area = 161; Cleaned area = 161

Noise level = 10 units; Cover time = 1341;

Rule: ANT_WALK_1 (no backtracking);

Figure 5: Three agents, oriented by the ANT-WALK-1 rule, with random sensory-noise of 10 units. This noise does not avoid the agents from fulfilling their mission; however a significant delay is introduced.



I. A. Wagner

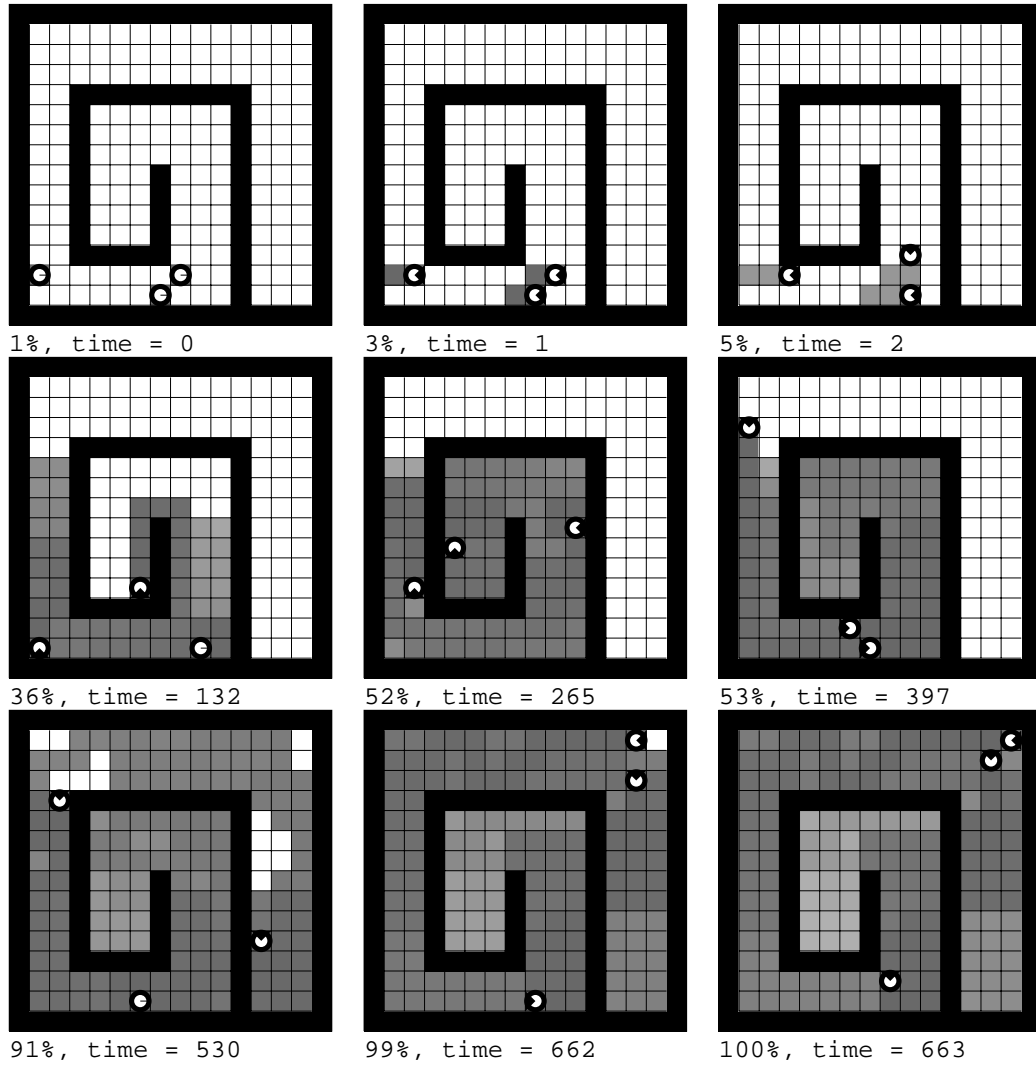
Covering by ANT-WALK-2, shape #3, on a 14 x 14 matrix

Num of Ants = 3; Total area = 161; Cleaned area = 161

Noise level = 0 units; Cover time = 691;

Rule: ANT_WALK_2 (backtracking);

Figure 6: Three agents, oriented by the ANT-WALK-2 rule, which is a generalization of the DFS algorithm. Covering here is much more efficient (27% of the time required by ANT-WALK-1 rule) due to its backtracking ability.



I.A. Wagner

Covering by ANT-WALK-2, shape #3, on a 14 x 14 matrix

Num of Ants = 3; Total area = 161; Cleaned area = 161

Noise level = 10 units; Cover time = 663;

Rule: ANT_WALK_2 (backtracking);

Figure 7: Three agents, oriented by the ANT-WALK-2 rule, with sensory-noise level of 10 units. Note that even with noise, this algorithm is much faster than ANT-WALK-1; it is also advantageous over the ordinary DFS which is extremely vulnerable to sensing errors.

rule	noise	#robots:	1	2	3	4	5	6	7	8	9	10
ANT-WALK-1	0		1159	325	550	320	320	307	143	275	123	123
	10		2721	836	1341	573	883	568	267	653	501	484
	20		13461	2707	2402	2375	1656	513	836	972	265	262
ANT-WALK-2	0		1038	1219	691	270	270	270	266	183	60	60
	10		2472	517	663	836	546	576	652	581	211	270
	20		4105	1349	1079	951	840	734	613	383	370	165
VERTEX-ANT-WALK	0		445	252	172	140	118	125	88	123	69	69
	10		6965	2478	2060	1474	1469	1212	1035	1010	863	734
	20		12839	6222	4320	3212	2768	1727	1681	1691	1671	1322

Figure 8: Covering times for the three algorithms running on a 14×14 maze-shape with a varying amount of noise. Note that the time is not always monotonically decreasing; in some cases additional robots disturb the efficiency of the whole group.

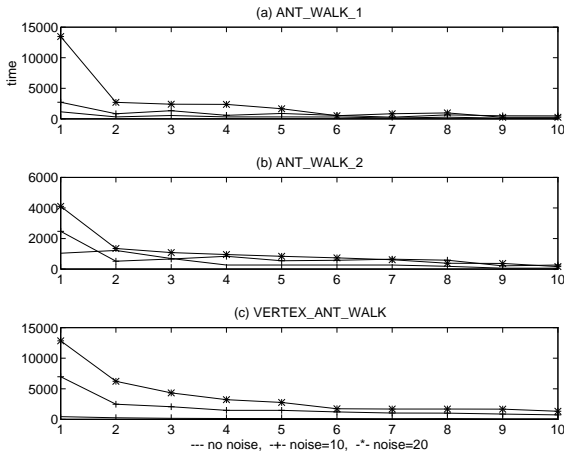


Figure 9: Time of covering, t_k , versus k - the number of robots, with varying amount of sensing-noise, for the three ANT-WALK algorithms. Note the different scales on the vertical axis. It can be seen that ANT-WALK-2 has best performance in the given test case.

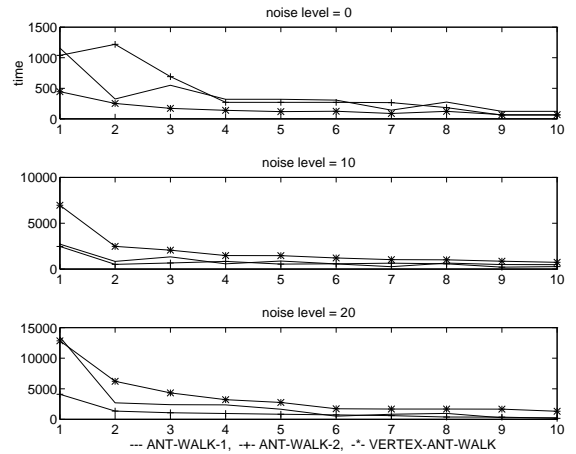


Figure 10: The same data of the previous figure is here sorted by noise-level. It clearly shows the VERTEX-ANT-WALK is best in the absence of noise, while ANT-WALK-2 wins in noisy conditions.

rule	noise	#robots:	1	2	3	4	5	6	7	8	9	10
ANT-WALK-1	0		456	166	135	105	84	88	55	52	43	36
	10		2673	1254	955	636	555	507	457	324	317	263
	20		3802	1987	1230	963	856	771	531	473	413	419
ANT-WALK-2	0		884	480	356	185	150	127	108	118	77	66
	10		3396	791	591	506	377	279	235	200	167	194
	20		3855	897	649	478	454	446	302	254	243	235
VERTEX-ANT-WALK	0		660	312	239	169	147	118	110	93	74	65
	10		3710	1083	647	395	374	339	269	331	262	231
	20		4646	1991	1228	836	696	660	423	480	300	372

Figure 11: Covering times for the three algorithms running on a 14×14 maze-shape with a varying amount of noise, averaged on 10 coverings; The table shows that on the average, more robots bring a faster covering, with some exceptions caused by the random nature of the noise.

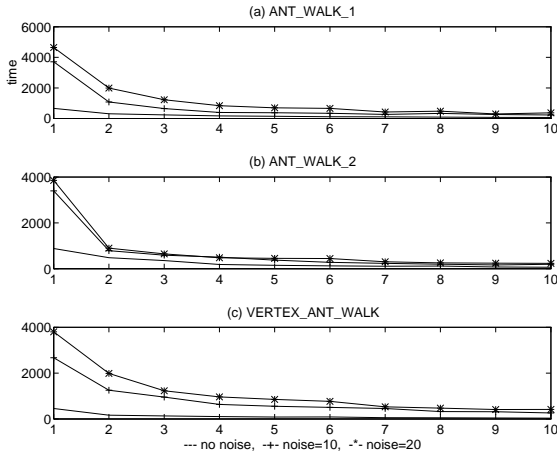


Figure 12: Time of covering, t_k , versus k - the number of robots, with varying amount of sensing-noise, for the three ANT-WALK algorithms, averaging on 10 coverings.

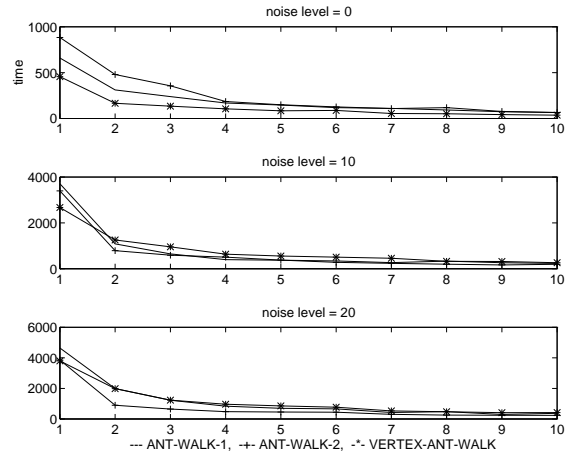
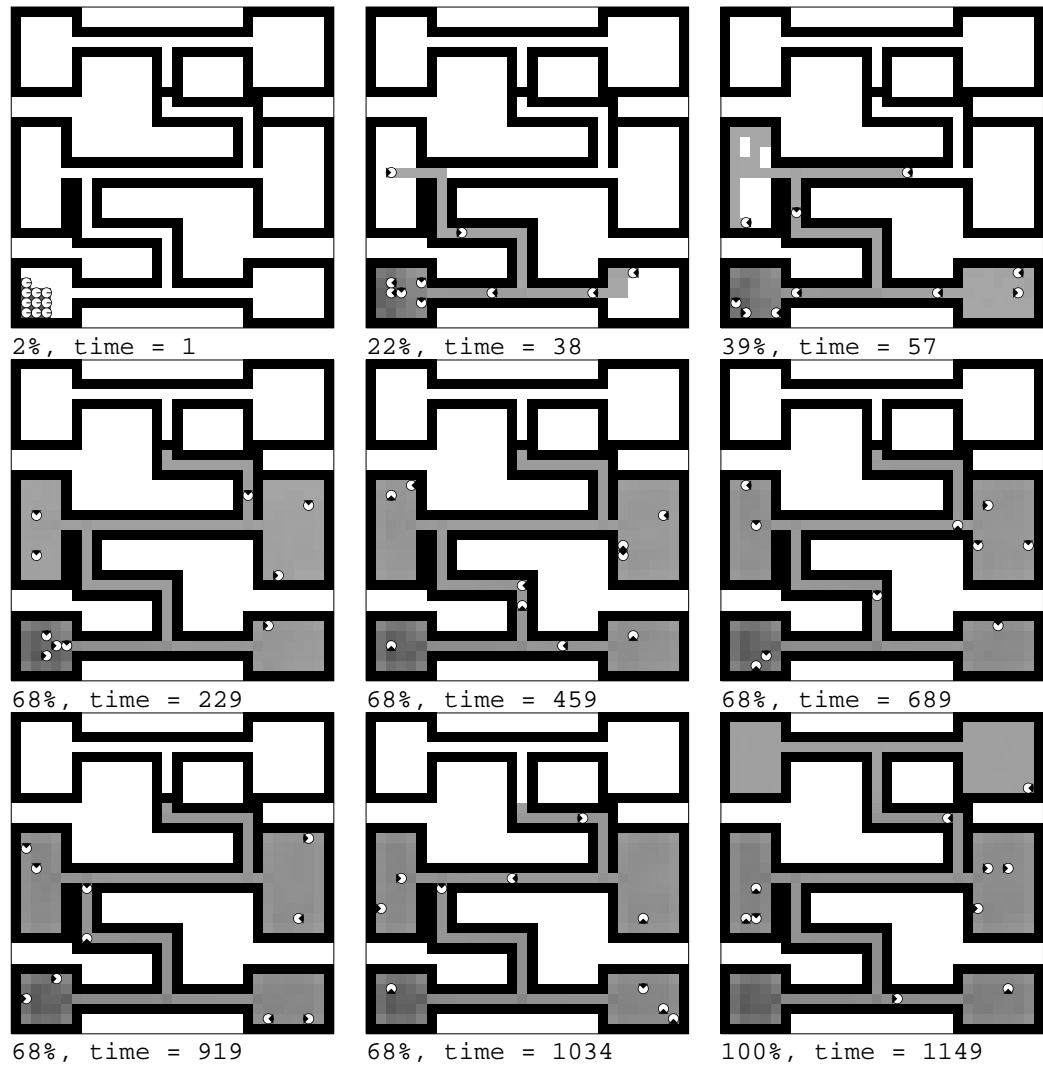


Figure 13: The same data (average of 10 covers) of the previous figure, sorted by noise-level. Here again ANT-WALK-2 wins with noisy sensors.



Covering by ANT-WALK, shape #5, on a 30 x 30 matrix

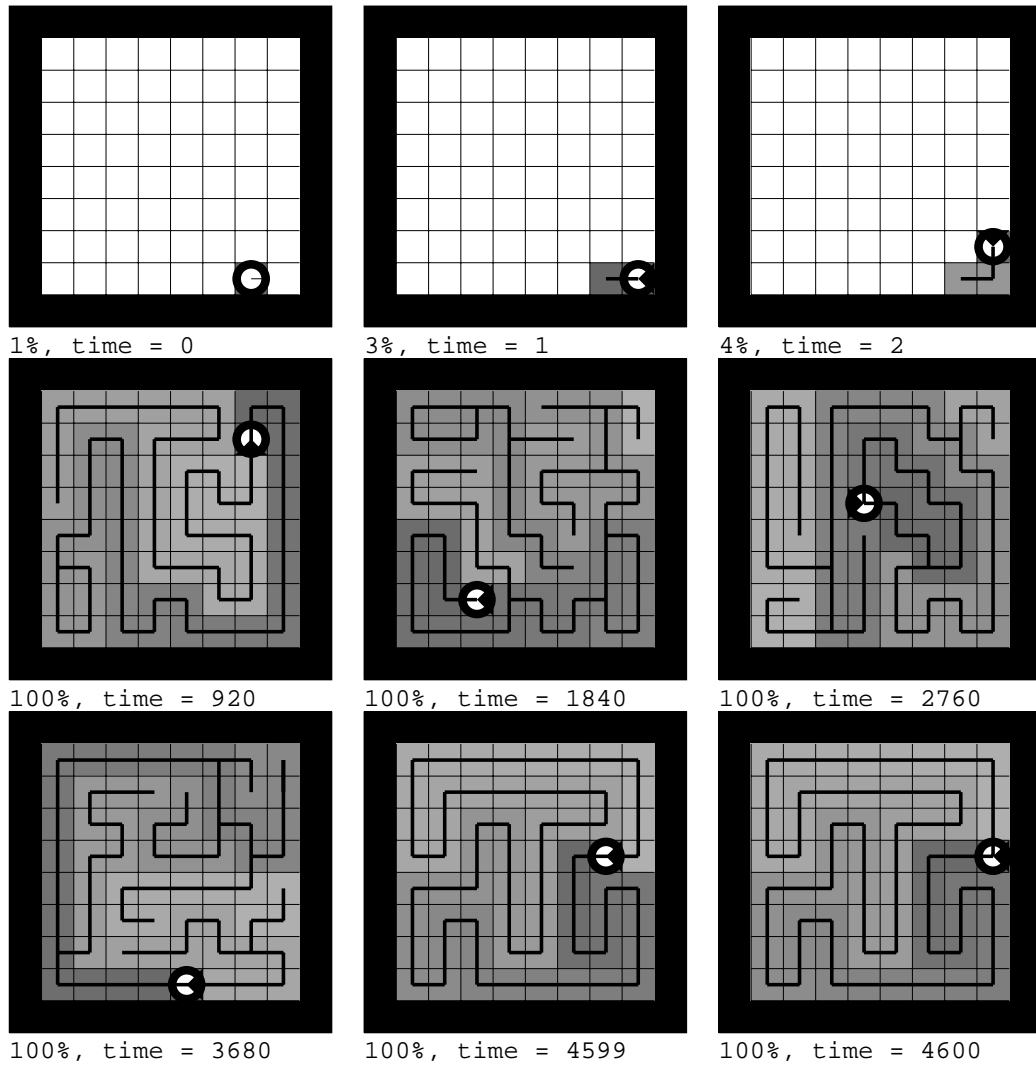
Num of Ants = 10; Total area = 338; Cleaned area = 338

Noise level = 0 units; Cover time = 1149;

Rule: ANT_WALK_2 (backtracking);

Region changed at time 1000;

Figure 14: Ten ants overcoming a change in the environment, by doing a multi-level DFS (ANT-WALK-2). It demonstrates the ability of the algorithm to cope with a dynamic environment.



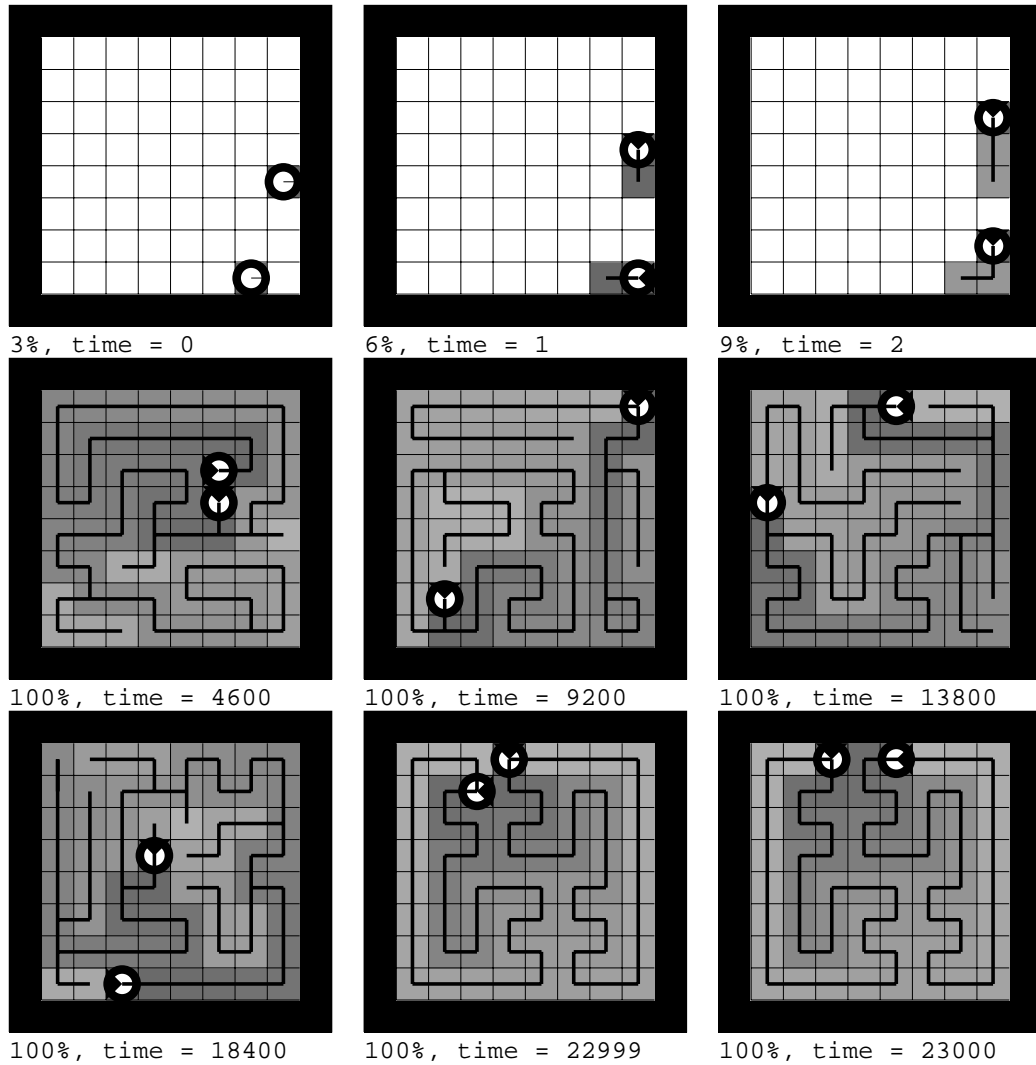
Covering by VERTEX-ANT-WALK, shape #0, on a 8 x 8 matrix

Num of Ants = 1; Total area = 64; Cleaned area = 0

Noise level = 0 units; Cover time = 0;

Heuristic: (visits, time);

Figure 15: The VERTEX-ANT-WALK dynamics has the interesting property that cycle covers of the graph are among the limit cycles. Here a Hamiltonian path (i.e., a special case of cycle-cover) was found as a limit cycle of the process. The black lines describe the edges traversed by the ant in the most recent n units of time.



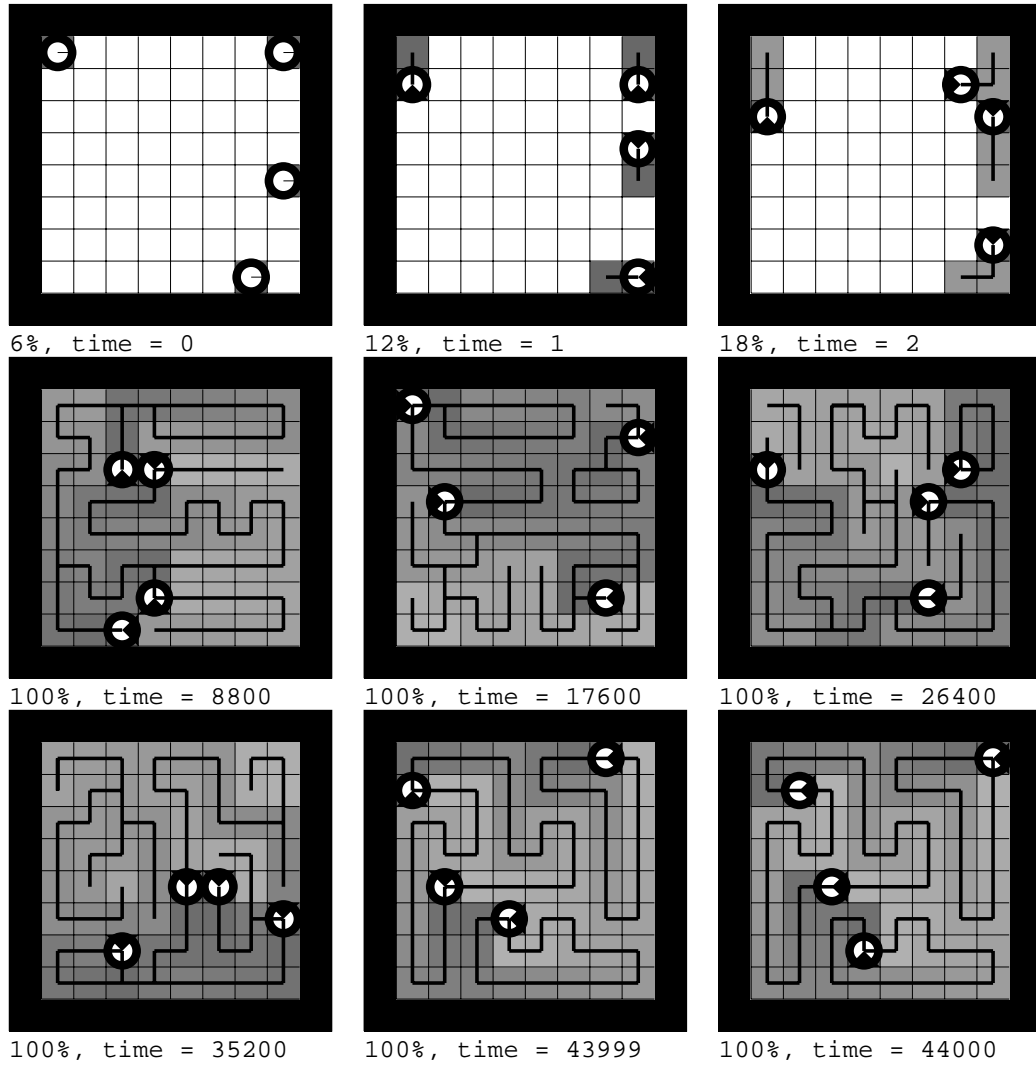
Covering by VERTEX-ANT-WALK, shape #0, on a 8 x 8 matrix

Num of Ants = 2; Total area = 64; Cleaned area = 0

Noise level = 0 units; Cover time = 0;

Heuristic: (visits, time);

Figure 16: Here again a cycle-cover is found as a limit cycle of the VERTEX-ANT-WALK process; this time we get, in the limit, two cycles covering the graph of tiles.



I. A. Wagner

Covering by VERTEX-ANT-WALK, shape #0, on a 8 x 8 matrix

Num of Ants = 4; Total area = 64; Cleaned area = 0

Noise level = 0 units; Cover time = 0;

Heuristic: (visits, time);

Figure 17: Same region as in the previous two figures, but this time with 4 ants; a triple cycle-cover of G is achieved in the limit. An open question is whether or not the time of convergence to such a limit-cycle can be reasonably bounded.

References

- [1] F.R. Adler, D.M. Gordon, "Information Collection and Spread by Networks of Patrolling Ants," *The American Naturalist* **140** No. 3 Sept. 1992.
- [2] R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovasz, C. Rakoff, "Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems," in *20'th Annual Symposium on Foundations of Computer Science*, p. 218-223, San Juan, Puerto Rico, October 1979.
- [3] W. Alt, "Biased Random Walk Models for Chemotaxis and Related Diffusion Approximations," *J. Math. Biology* **9**, 147-177 (1980).
- [4] C.M. Bender, S.A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers*, McGraw-Hill, 1978.
- [5] 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.
- [6] B. Bollobas, *Graph Theory - an Introductory Course*, Springer-Verlag, 1990.
- [7] V. Braitenberg, *Vehicles*, MIT Press 1984.
- [8] R.A. Brooks, "Elephants Don't Play Chess," in *Designing Autonomous Agents*, P. Maes (Ed.), pp. 3-15, MIT Press/Elsevier, 1990.
- [9] R.A. Brooks, "Intelligence without representation," *Artificial Intelligence*, 47 (1991) 139-159, Elsevier.
- [10] M. Blum, W.J. Sakoda, "On the capability of finite automata in 2 and 3 dimensional space," in *FOCS'77*, p. 147-161.
- [11] G. Barnes, U. Feige, "Short Random Walks on Graphs," in *Proc. of the 25'th ACM STOC*, 1993.
- [12] M. Blum, D. Kozen, "On the power of the compass, or, Why mazes are easier to search than graphs," in *FOCS'78*, p. 132-142.
- [13] A.Z. Broder, A.R. Karlin, P. Raghavan, E. Upfal, "Trading Space for Time in Undirected $s-t$ Connectivity," *SIAM J. COMPUT.*, Vol. 23, No. 2, pp. 324-334, April 1994.
- [14] R.A. Brooks, "Artificial Life and Real Robots," in *Proc. of the First European Conference on Artificial Life*, MIT Press/Bradford Books, Cambridge, MA, 1992, pages 3-10.
- [15] A.M. Bruckstein, "Why the Ant Trails Look So Straight and Nice ," *The Mathematical Intelligencer*, vol. 15, No. 2, pp. 59-62, 1993.
- [16] 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.
- [17] S.H. Clearwater, B.H. Huberman, T. Hogg, "Cooperative Problem Solving," *SCIENCE*, VOL. 254, Nov. 1991, pp.1181-1183.

- [18] S.H. Clearwater, T. Hogg, B.H. Huberman, "Cooperative Solution of Constraint Satisfaction Problems," in *Computation: the Micro and the Macro View*, B.A. Huberman (Ed.), pp. 33-70, World Scientific, 1992.
- [19] Z. Collin, S. Dolev, "Self-stabilizing depth-first search," *Information Processing Letters*, 49 (1994) 297-301.
- [20] C. Cookson, "A Brave New Olfactory Bulb," *Financial Times*, Thursday, June 8 1995, p. 10.
- [21] J.L.Deneubourg, S.Aron, S. Goss, J.M. Pasteels, G. Duerink, "Random Behaviour, Amplification Processes and Number of Participants: How They Contribute to the Foraging Properties of Ants," *Physica* 22D (1986) 176-186.
- [22] E.W. Dijkstra, "Self-stabilizing Systems in Spite of Distributed Control," *Comm. ACM*, 17 (1974) 643-644.
- [23] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, Maryland, 1979.
- [24] A.S. Fraenkel, "Economic Traversal of Labyrinths," *Mathematics Magazine*, 43: 125-30, 1970, and a correction in 44: 12, 1971.
- [25] S. Gal, E.J. Anderson, "Search in a Maze," *Probability in the Engineering and Informational Sciences*, 4, 1990, 311-318,
- [26] B.K. Holldobler, E.O. Wilson, "Weaver Ants," *Scientific American*, 237(6), pp. 146-154, 1977.
- [27] J. Huxley, *Ants*, Arrow Books, London, 1962.
- [28] J. Hopcroft, R. Tarjan, "Efficient Algorithms for Graph Manipulation," *Comm. ACM* , June 1973, pp. 372-378.
- [29] S. Hedberg, "Robots Cleaning Up Hazardous Waste," *AI Expert* , May 1995, pp. 20-24. Springer-Verlag 1994.
- [30] A. Itai, C.H. Papadimitriou, J.L. Szwarcfiter, "Hamilton Paths in Grid Graphs," *SIAM J. on Computing* (1982), 11:676-686
- [31] *Instantiating Real-World Agents*, Papers from the AAAI 1993 Fall Symposium, Tech.Rep. FS-93-03, AAAI Press, Menlo Park, California.
- [32] M. Kac, "Some Mathematical Models in Science," *Science* 166 (1969), 695-699.
- [33] S. Katz, O. Shmueli, "Cooperative Distributed Algorithms for Dynamic Cycle Prevention," *IEEE T-Software Eng.* Vol. SE-13, No. 5, May 1987.
- [34] S. Lefschetz, *Differential Equations: Geometric Theory*, Interscience Publishers, New York, 1957.
- [35] X. Lin, P.K. McKinley, L.M. Ni, "The Message Flow Model for Routing in Wormhole-Routed Networks," *IEEE T-Par. and Dist. Sys.*, Vol. 6, No. 7, July 1995.
- [36] L.M. Ni, P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE COMPUTER* magazine, February 1993.

- [37] E.M. Rauch, M.M. Millonas, D.R. Chialvo, "Pattern Formation and Functionality in Swarm Models," to appear in *Physics Letters A*, 1995.
- [38] P. Sachs, "Transforming Work: Collaboration, Learning, and Design," *Communications of the ACM*, Sept. 1995.
- [39] W. Neubauer, "Locomotion with Articulated Legs in Pipes or Ducts," *Robotics and Autonomous Systems*, 11:163-169. Elsevier 1993.
- [40] J.D. Nicoud, M.K. Habib, "The Pemex B autonomous demining robot: perception and navigation strategies," *Proc. 1995 IEEE/RSJ Int'l conf. on Intelligent Robots and Systems, Human-Robot Interaction and Cooperative Robots*, Pittsburgh, PA, Aug. 1995, vol. 1, pp. 419-424.
- [41] R.A. Russell, "Laying and Sensing Odor Markings as a Strategy for Assisting Mobile Robot Navigation Tasks," *IEEE Robotics and Automation magazine* , Vol. 2, No. 3, 1995, pp. 3-9.
- [42] R.A. Russell, "Mobile robot guidance using a short-lived heat trail," *Robotica* , Vol. 11, 1993, pp. 427-431.
- [43] Y. Shoham, M. Tennenholtz, "On Social Laws for Artificial Agent Societies: Off Line Design," to appear at *AI-Journal*, 1995.
- [44] H.A. Simon, *The Sciences of the Artificial*, 2nd ed., MIT Press, 1981.
- [45] S. Sen, M. Sekaran, J. Hale, "Learning to Coordinate Without Sharing Information," *Proceedings of AAAI-94*, pp. 426-431.
- [46] L. Steels, "Cooperation Between Distributed Agents Through Self-Organization," *Decentralized A.I. - Proc. of the 1st European Workshop on Modeling Autonomous Agents in Multi-Agent World*, Y. DeMazeau, J.P. Muller (Eds.), pp. 175-196, Elsevier, 1990.
- [47] R. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM J. Comput.*, vol. 1 no. 2 (1972), pp. 146-160.
- [48] G. Tarry, "Le problem des labyrinths," *Nouvelles Annales de Mathematiques*, 14:187.
- [49] I.A. Wagner, A.M. Bruckstein, "Cooperative Cleaners - a Study in Distributed Ant-Robotics," *Proceedings of the 3rd French-Israeli Symposium on Robotics*, Herzlia, May 1995. (A later version was presented at the *1st Online Workshop on Evolutionary Computation*, Nagoya university and the WWW, October 1995.) The complete version is in *Technical report CIS-9512, Center for Intelligent Systems*, Technion, Haifa, June 1995.

Appendix

Using the ANT-WALK Covering Method to Maintain a self-Stabilizing Spanning Tree in a Dynamic Network

The system of ants and smell-traces is adaptive and will change to comply with changes in the environment, as demonstrated in Figure 14. A common notion of adaptivity in distributed algorithms is self-stability. A distributed algorithm is *self-stabilizing* (as defined in [19]) if it can be started from *any possible* global state and once started, the algorithm regains consistency by itself. The study of such algorithms started with [22]. In [19], A self-stabilizing algorithm is described that maintains a DFS tree in a network of processors, and guarantees recovery from a topological change in the graph within time $O(nd\Delta)$.

Our system of k cleaning agents is self-stabilizing since it will recover from any change in the smell levels on the edges, (e.g. if a wind-blow has scrambled the smell-traces) or even a change in the topology of the graph, as long as the graph remains connected. This property does not exist in traditional search algorithms like Depth-First-Search and Breadth-First-Search (see, e.g. [23]), since those methods rely on the absolute marking of the edges, while our method only uses the marks as relative quantities.

Now let us assume that a network is given in which nodes and edges occasionally become ineffective, and we need to keep a distributed tree that spans the network, i.e. - each (effective) non-root node should know who his “father” node is in the tree. If our agents have distinct *id*’s, say $1, 2, \dots, k$, then they can leave a signature at each vertex in the form of a pair (t, id) , where t is the last time an agent visited the vertex, and id is his unique identification number. Clearly, after our agents have covered the graph, no two vertices will have the same signature. Hence a spanning tree can be established by having each node taking his greatest neighbor as a father; here “greatest” means under the lexicographic order of the pairs (t, id) among the neighbors. If a node or an edge is crashing, we are guaranteed (by Theorem 1, 2 or 3) that after no more than (the respective) t_k units of time the tree will recover.

The advantage of the above algorithm over existing methods for keeping cycle-free communication graphs (e.g. [33], [19]) is that in our method only k of the n processors need to work at a time - the others can proceed in their regular jobs.