Smell as a Computational Resource - A Lesson We Can Learn from the Ant

Israel A. Wagner
Department of Computer Science, Technion
and
IBM Haifa Research Lab, Matam, Haifa 31905, Israel
wagner@haifasc3.vnet.ibm.com

Michael Lindenbaum
Department of Computer Science, Technion
mic@cs.technion.ac.il

Alfred M. Bruckstein
Department of Computer Science, Technion
(currently on sabbatical at
Bell Laboratories, Murray-Hill, NJ 07974, USA)
freddy@cs.technion.ac.il

Abstract

Some insects are known to use chemicals called pheromones for various communication and coordination tasks. In this paper we follow an ancient advice [1] and 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 multia(ge)nt 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. A similar smell-oriented mechanism can be used to keep a spanning tree of a dynamic network.

[1] "Go to the ant, thou sluggard; consider her ways, and be wise" (Proverbs,vi,6).

1. Introduction

One of the basic theoretical (and practical) problems in multiagent systems (see, e.g., [5]) is how to design adaptive rules of behavior for the individual, that will lead to the required emergent results while reducing cost in terms of communication overhead and hardware complexity. We consider a cleaning task in which a building has to be cleaned by a group of autonomous robots who do not have a prior knowledge of the building's structure. Our assumption is that the floor in the building constitutes of small tiles, all of the same size, and that a tile can be cleaned in one unit of time. In order to help their navigation, the robots are allowed to leave traces along their walk, e.g. by means of smell, heat, or color trails. We assume that the intensity of traces decreases with time, and by comparing the smell levels at two neighbor tiles, the robot can deduce which tile was visited more recently. We shall later use this property to navigate efficiently among the tiles. See Figure 1 for an example, and note that our assumption is that the robots are small enough such that several robots can occupy the same tile simultaneously. The topology of the building may change during the robots work; e.g. people or furniture may move and doors may open or close, hence a preliminary phase of mapping will not be of help here. 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 a local, cue-based operation that solves the above problems, and

analyze their worst-case performance. 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, and bound their worst-case time complexity.

As far as we know, it is the first analyzable model that considers the *cooperative* potential of trace-oriented behavior in terms of time-complexity. Our model is different from the others we are aware of, in that we assume that the smell is both created and used by the same group of foraging ants.

This model can be used as a way to understand ants behavior, but can also inspire the designers of multi-agent robotic systems. Some of the potential applications of the smell-oriented walking model are in cooperative cleaning of a dirty region with obstacles, and in the maintenance of spanning tree in a communication network.

Our **main results** are upper bounds on the cover time of the three methods for trace-based search. The first is ANT-WALK-1, in which no individual memory is assumed on each machine, the only (shared) memory being the smell traces that are being laid on the edges of G. Denoting by t_k the time needed to cover the edges of the graph by k agents that follow this rule, and in the presence of α units of sensing-noise, the following upper bound on this time is proved:

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, $n = |V(G)|$ 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 famous 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 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.

Related Work: Graph search is an old problem; several methods exist for deterministic (e.g. [25], [15]) random (e.g. [2], [7], [8]) and semi-random ([16]) covering, but a lot more needs to be done in order to make the theory useful in the context of robotic covering problems. 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, even if its vertices are on the grid, is NP-complete even if the graph is completely known [19]. Some aspects of the behavior of continuous smell-oriented swarms were investigated in [22], 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 distinguished and investigated in several works, e.g. [17] and [12]. Odor marking as an assistance in robot covering and navigation has been presented in [23], where the structure of smell sensors is described. Unfortunately the geometrical theory of multi-agent systems is far from being satisfactory, as has been pointed out in [5] and many other papers. In [26] 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.

The rest of the paper is organized as follows: in the next section we state the problem formally, and present the algorithm ANT-WALK-1 with its analysis. Section 3 is devoted to an improved algorithm, ANT-WALK-2, and section 4 - to a vertex oriented method of search. In Section 5 we discuss the use of trace-oriented walks for creating and maintaining a spanning tree in a dynamic network, then show some simulation examples in Section 6, and conclude with a summary and a discussion in Section 7.

2. ANT-WALK-1: Covering Without Individual Memory

Consider a connected, undirected graph $G = (V, E)$. 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 vertex, coded by some trace left by the agent. In this rule of motion, an ant visiting vertex $u \in V(G)$ checks all edges emanating from u. Then it goes to an edge that has the oldest trace on it, that is - the edge that was most previously visited. In the

course of traversing the edge, the ant leaves there a constant amount of pheromone. The amount of smell on a directed 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. Unfortunately, this ability is limited by the sensing error α - in the presence of such an error, one can only distinguish between two traces that differ by more than α. Hence, leaving a smell trace on (u, v) at time t is similar to writing the number t on this edge. For sake of simplicity, we'll denote traces by the time they were left. It is assumed that a new trace is overriding an older trace left on the same passage. 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 $N(u)$.

2.1. A Rule of Motion

Formally, the first 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 order of exiting the tile is determined by the priority between the agents. This order of priorities can be determined either by a unique hard-coded ID, 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.

2.2. Analysis of the ANT-WALK-1 Algorithm

In this subsection we'll prove an upper bound on the time complexity of the ANT-WALK-1 algorithm, assuming $\alpha = 0$. Noise will be considered in the next subsection. The basic idea in our proof of the upper bound on the cover time is to use network-flow theory to show that the number of passages along one edge cannot differ too much from the number of passages on any other edge in the graph. Let us define $f(x, y)$, the *flow* along edge $(x, y) \in E$, as the number of times any robot went over this edge in the direction from x to y. The following lemma says that 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 outgoing flow from any vertex is fairly distributed among its neighbors.

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

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

Proof: The assertion of the lemma is an invariant - it holds at time $t = 0$ (when for all edges $(x, y) \in E$, $f(x, y) = 0$), and remains true since an agent always selects an emanating edge with the oldest trace on it (step B in the ANT-WALK-1 rule) which is also the edge with minimum number of visits on it. This causes that the order of visiting the edges of a given vertex is always the same. □

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

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

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

Lemma 2 *At all times, and for all cuts $(S : \overline{S})$ in G,*

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

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 $(S : \overline{S})$, 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:

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

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

We shall now combine the fairness of flow (Lemma 1) and its boundedness (Lemma 2) to show that the intensity

of flow cannot differ too much between adjacent vertices in G. This will later lead us to conclude that the process has a good "mixing property" over time.

For each vertex $x \in V$ let $g(x)$ denote the maximum flow along an edge emanating from x

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

Recall that due to Lemma 1, all edges (x,y) emanating from x have flow $f(x,y)$ such that $g(x) - 1 \leq f(x,y) \leq g(x)$. Now order the vertices of V in increasing order of $g(.)$, i.e.

$$g(x_1) \leq g(x_2) \leq \ldots \leq g(x_n)$$

where n is the cardinality of V. Also denote by $S[i,j]$ the set $\{x_i, x_{i+1}, \ldots, x_j\}$, and by $C(i,j)$ the cut between $S[1,i]$ and $S[j,n]$. i.e.

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

In the following lemma we show that the function $g(.)$ has a discrete "smoothness" property which is similar to the *Lipschitz Continuity* [2] property for real functions.

Lemma 3 *At all times and for all* $1 \leq i < j \leq n$

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

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. $p \leq i, \quad q \geq i+1$) 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:

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

But this net flow is also bounded above by k (Lemma 2), so we get that

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

and hence

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

which yields the lemma. □

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

[2] A function $f(x)$ is *Lipschitz Continuous* with coefficient δ if

$$\forall x,y : \quad |f(x) - f(y)| \leq \delta |x - y| .$$

See [4],[21] for generalizations and applications of Lipschitz continuity

Corollary 1

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

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 [3], it can easily be seen that

Corollary 2

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

A more sophisticated (and in general, smaller) bound on the difference in $g(.)$ between vertices can be achieved by defining $\rho(G)$, the *cut-resistance* of graph G, as follows:

Definition 1 *Assume that $\{x_1, \ldots, x_n\}$ 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*

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

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

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

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

$$\rho(G) = \sum_{i=1}^{n-1} \frac{1}{i(n-i)} < 1.$$

- if G is an $m \times m$ grid-graph then $\rho(G) = O(m) = O(\sqrt{n})$.
- in general, $\rho(G) < (n-1)/\lambda(G)$, 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 3

[3] The 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.

Corollary 3

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

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

Corollary 4

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

Remark: Please note that $\rho(G)$ is better suited for our purposes than the edge-expansion factor [4] 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 than that of the complete graph, since almost any cut has at least $n/2$ edges in it.

Let us denote by $t_k(G)$ 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:

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

(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. □

Remark 1: The bound we proved is nearly tight (i.e. 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 edge set. However, such problems can be cured using a good heuristic in resolving 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 sequel.

Remark 2: The expression in parentheses, $\left(\frac{n}{k} + \rho(G)\right)$, can be explained intuitively as follows: If the graph is dense, its cut-resistance $\rho(G)$ is low 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, as exemplified in our simulations.

2.3. The Effect of Noise and Sensing-Errors on the Dynamics of ANT-WALK-1

In reality, sensors and effectors are not always perfectly reliable and noise can disturb their operation. In [9] a rather skeptical statement was made:

> ...[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 the individual agents. 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

[4] The *edge expansion factor* of a graph G is defined as

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

only on relative quantities which do not suffer so much from the errors in sensor readings.

As a preliminary model of noise, we consider a global noise parameter α, that represents a bound on the amount of incorrectness in the reading of a sensor. This means that if the smell along edge (x, y) is equal to $s(x, y)$, the noisy sensor may tell us anything between $s(x, y) - \frac{\alpha}{2}$ and $s(x, y) + \frac{\alpha}{2}$. 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. The new "noisy" form of Lemma 1 is

Lemma 1.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,

$$|f(u, v) - f(u, w)| \leq \alpha + 1. \qquad (2)$$

Consequently, Lemma 3 becomes

Lemma 3.α *With sensor deviation at most α, it always holds that for all $1 \leq i < j \leq n$*

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

Finally we get a modified version of Theorem 1:

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.

It is interesting to observe here that as the noise parameter α grows, there is more sense in using a larger number of robots. It is in compliance with the experimental results presented in [10] (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.

3. ANT-WALK-2: Covering with Backtracking - a Multilevel Version of the DFS Algorithm

3.1. A Second Rule of Motion

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 is essentially a generalization of the famous Depth-First-Search algorithm. However, there are some problems in using the conventional DFS (as in [24],[18],[14]) as is:

1. Does not give itself to cooperation - once a robot got back to its starting point, it will (according to DFS) stop there forever, rather then go around and help his hard-working fellows.
2. DFS is critically sensitive to errors in sensing the marks it left on the edges - once a "forward edge" is erroneously interpreted as "backward edge" - it may get stuck and fail to cover the graph.
3. If a change occurs in G (e.g. an edge is deleted) that does not disconnect G, but interrupts the DFS backtracking path, then DFS is getting stuck, being unable to cover the graph.

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 (individual) **search-level**(r) variable. This variables stores the time when "the new history begins" i.e. any edge/vertex visited before that time is considered un-visited by robot r. 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 [14]), 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.

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 then 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 variables 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 $t_{in}(u) <$ **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. $s(u,v) <$ **search-level**(r)
then
set $s(u,v) := t$;
if $t_{in}(v) <$ **search-level**(r) go to v;
/* have I exhausted the current level of search ? */
C) if $t_{in}(u) > t_{out}(u) \geq$ **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.

3.2. 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, and, as proved in [14], covers the graph in time at most $2|E|$, then starts a new tour and so on.

For more then 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 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 2.3. Then in the worst case, α levels of search are needed to guarantee a full coverage of the graph. Hence we get

Theorem 2 *k agents obeying* ANT-WALK-2 *and having noise error level of at most α 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 taversed 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 $\lceil \frac{1+\alpha}{k} \rceil$ times t_1 - the cover time of one noise-less robot. □

4. VERTEX-ANT-WALK - A Vertex-oriented search

In this variation we consider the problem of visiting all vertices of a graph, rather than all edges as before. For this purpose we use smell traces that are being laid on the vertices of the graph and will be represented by the functions $s(u)$ for all $u \in V(G)$. Being in vertex u, an agent assigns the current value of his clock to the variable $s(u)$, and then goes on to the neighbor of u with least t-value. Formally, the rule is :

> 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 pheromone on u */
> C) set $s(u) := t$;
> D) go to v;
> end VERTEX-ANT-WALK.

A system of agents obeying this rule is guaranteed to cover the vertices, and works quite well in simulations. The upper bound on we have on its cover time is exponential in the diameter of the graph:

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, where*

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

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 grater than $s(u)$, hence u should be visited no later than after the next visit to v. Hence we get

$$f(u) \leq \Delta(f(v) + 1) \tag{3}$$

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 $P = x_1, x_2, \ldots, x_q$. Clearly q, the length of the path, is smaller than or equal to d, the diameter of G. Using Equation 3 we get

$$\begin{aligned} f(x_q) &\leq \Delta + \Delta f(x_{q-1}) \leq \Delta + \Delta^2 + \Delta^2 f(x_{q-2}) \leq \ldots \\ &\leq \Delta^q + f(x_1) \leq \Delta^d + f(x_1). \end{aligned}$$

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

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

□

Please note our assumption that initially all agents are located on distinct vertices, hence when $k \to \infty$ 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 will eventually be discovered.

The dynamics involved with this vertex-orient ant-walk has the interesting property of having cycle covers of the graph as its fixed points, i.e. once the k agents are in a loop of one (or more, up to k) cycle(s), they continue hoping on these cycles forever. In the case of a single agent, such a cycle is a Hamiltonian cycle; See Figure 5. For a $k > 1$, its a 2-factor (or "cycle cover") of the graph. Hence, such a dynamic may serve as a heuristic for finding a Hamiltonian path. Its efficiency has yet to be investigated.

5. Using the ANT-WALK to Maintain a Self-Stabilizing Spanning Tree in a Dynamic Network

The system of the ants and smell-traces is adaptive and will change to comply with changes in the environment, as demonstrated in Figure 6. A common notion of adaptivity in distributed algorithms is self-stability. A distributed algorithm is *self-stabilizing*, (as defined in [11]) 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 [13]. In [11], 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 it remains connected. This property does not exist in traditional search algorithms like Depth-First-Search and

Breadth-First-Search (see, e.g. [14]), 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 $1, 2, \ldots, k$, 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 id. 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 our algorithm over existing methods for keeping cycle-free communication graphs (e.g. [20]) 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.

6. Simulations and Experiments

The two ANT-WALK algorithms were implemented in the C programming language on an IBM-RS/6000 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. Examples are shown in Figure 4. In this Figure, the gray level in a grid-point represents the amount of smell i.e. the time since last visit to this point. Figure 6 shows an example with a blocking 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. In Figures 7 and 8 we show the time of covering vs. the number of agents, with varying amount of noise for the two 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 either $s + p\alpha/2$, where p is a random number between -1 and 1.

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

7. Summary and 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.

This work suggests several possible directions for future research. e.g. considering a continuous smell-oriented walk, and introducing randomness into the robot's decisions. Another possible direction is to use this (or a similar) model to analyze the distribution of knowledge in an interconnected society of hypothetical academic creatures that communicate through their publications. Once such a creature has published a paper, it looks around for a new topic of interest, choosing the one with oldest publications among the topics related to the current one. This approach is justified by the wish to work on a topic which has not been studied recently (and hence has a greater potential for new discoveries), while utilizing the knowledge acquired in previous work. Our results show that obeying such a rule of behavior, assuming the universe of knowledge has a finite size, it will eventually be covered by those hypothetical academic creatures. (Note that "covering the universe" only means that each topic was sometime investigated, not that a single person knows everything; only the *emergent knowledge* is complete).

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

8. Acknowledgement

We wish to thank Shmuel Gal for his advice in search theory.

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] G. Barnes, U. Feige, "Short Random Walks on Graphs," in *Proc. of the 25'th ACM STOC*, 1993.

[8] 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.

[9] 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.

[10] S.H. Clearwater, T. Hogg, B.H. Huberman, "Cooperative Solution of Constraint Saisfaction Problems," in *Computation: the Micro and the Macro View*, B.A. Huberman (Ed.), pp. 33-70, World Scientific, 1992.

[11] Z. Collin, S. Dolev, "Self-stabilizing depth-first search," *Information Processing Letters*, 49 (1994) 297-301.

[12] 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* 22**D** (1986) 176-186.

[13] E.W. Dijkstra, "Self-stabilizing Systems in Spite of Distributed Control," *Comm. ACM*, 17 (1974) 643-644.

[14] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, Maryland, 1979.

[15] A.S. Fraenkel, "Economic Traversal of Labyrinths," *Mathematics Magazine*, 43: 125-30, 1970, and a correction in 44: 12, 1971.

[16] S. Gal, E.J. Anderson, "Search in a Maze," *Probability in the Engineering and Informational Sciences*,**4**, 1990, 311-318,

[17] B.K. Holldobler, E.O. Wilson, "Weaver Ants," *Scientific American*, 237(6), pp. 146-154, 1977.

[18] J. Hopcroft, R. Tarjan, "Efficient Algorithms for Graph Manipulation," *Comm. ACM* , June 1973, pp. 372-378.

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

[20] S. Katz, O. Shmueli, "Cooperative Distributed Algorithms for Dynamic Cycle Prevention," *IEEE T-Software Eng.* Vol. SE-13, No. 5, May 1987.

[21] S. Lefschetz, *Differential Equations: Geometric Theory*, Interscience Publishers, New York, 1957.

[22] E.M. Rauch, M.M. Millonas, D.R. Chialvo, "Pattern Formation and Functionality in Swarm Models," to appear in *Physics Letters A*, 1995.

[23] 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.

[24] R. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM J. Comput.*,vol. 1 no. 2 (1972), pp. 146-160.

[25] G. Tarry, "Le problem des labyrinths," *Nouvelles Annales de Mathematiques*, 14:187.

[26] I.A. Wagner, A.M. Bruckstein, "Cooperative Cleaners - a Case of Distributed Robotics," *Technical report, Center for Intelligent Systems*, Technion, Haifa, 1995.

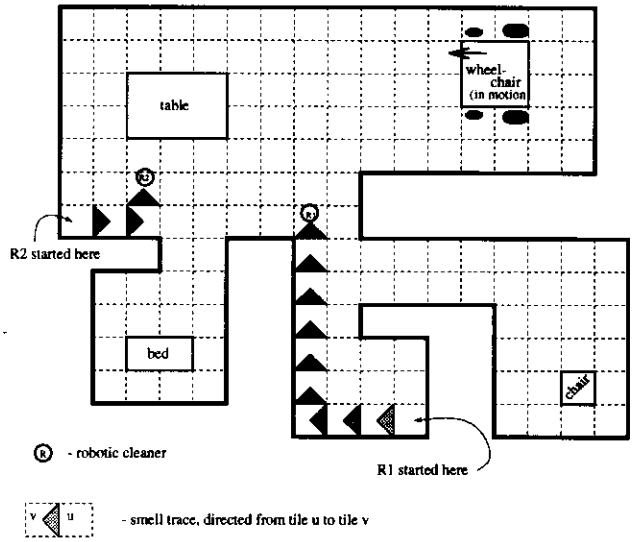

Figure 1. A system of rooms divided into square tiles. Two cleaning robots are shown with their diminishing smell traces.

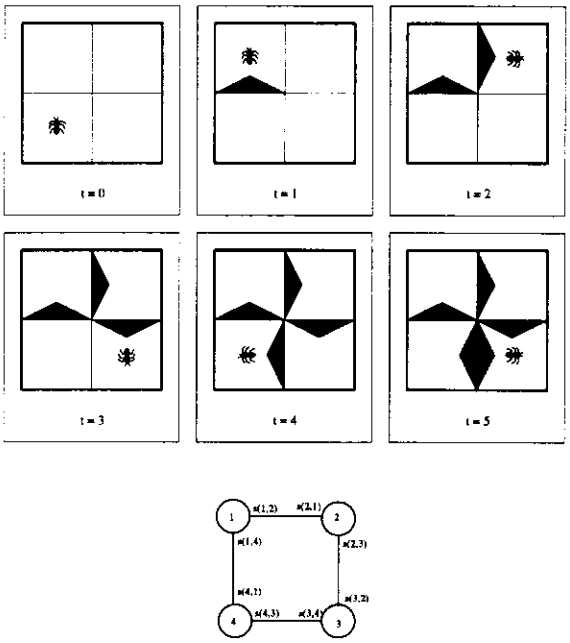

Figure 2. Four cycles of ANT-WALK-1 are needed to traverse a floor with 4 tiles. Also shown is the corresponding directed graph with the smell-labels on its edges. Note that there are two labels on each edge, to designate the trace intensity in each direction of the edge.

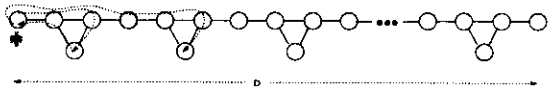

Figure 3. A hard case for the ANT-WALK-1 protocol. There are n vertices, $\approx 1.25n$ edges, the diameter is $\approx 0.8n$, and the time needed to traverse it may be as long as $O(n^2)$. The dotted arrows show the worst case where each triangle of vertices is a "trap" that causes the ant to go back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

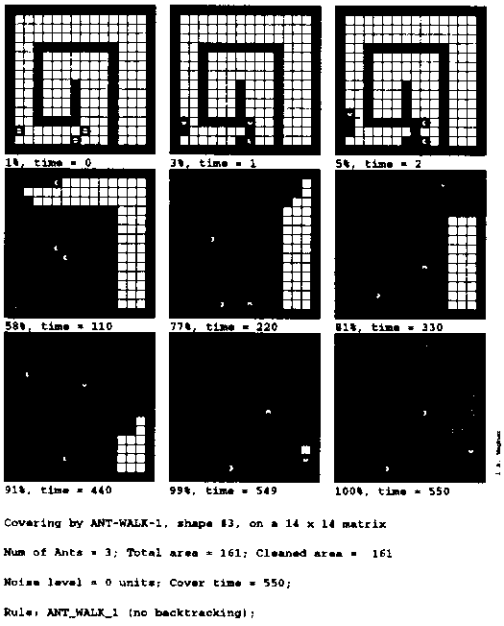

Figure 4. Three agents, oriented by the ANT-WALK-1 protocol. The gray-level is proportional to the number of visits in each point.

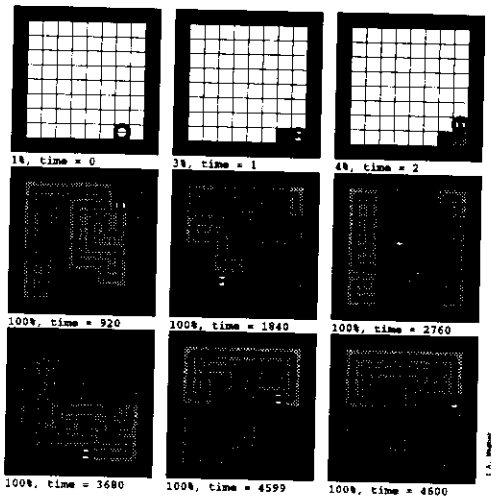

Figure 5. A Hamiltonian path may be found as a limit cycle of the VERTEX-ANT-WALK process. The black lines describe the edges traversed by the ant in the most recent n units of time.

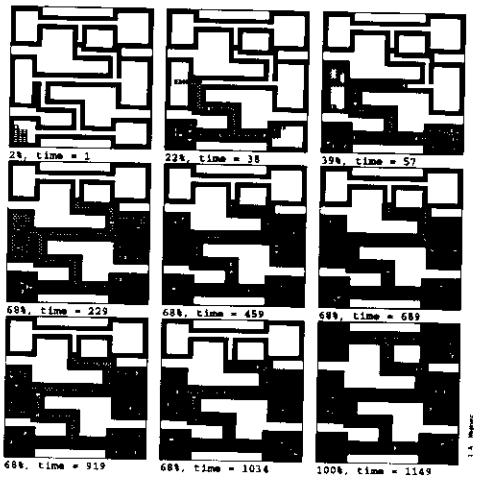

Figure 6. Ten ants overcoming a change in the environment, by doing a multi-level DFS.

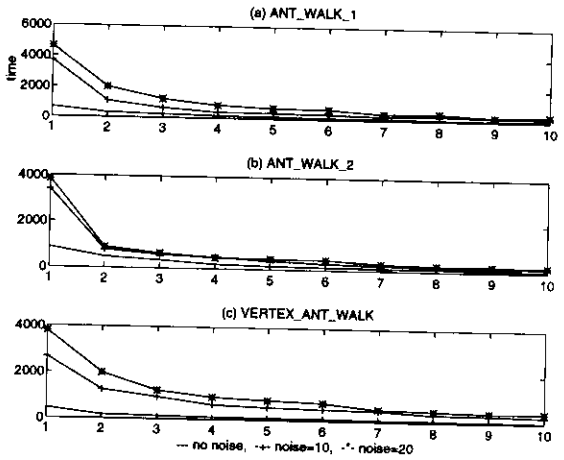

Figure 7. Time of covering , t_k, versus k- the number of robots, with varying amount of sensing-noise, for the three ANT-WALK algorithms. Note the different scales on the vertical axis. It can be seen that ANT-WALK-2 has best performance in the given test case.

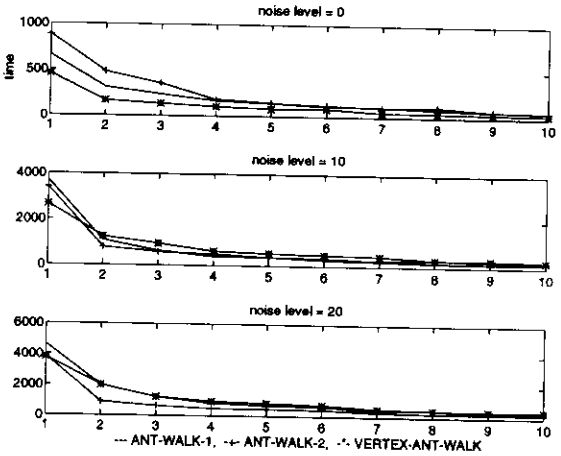

Figure 8. The same data of the previous figure is here sorted by noise-level. It clearly shows the VERTEX-ANT-WALK is best in the absence of noise, while ANT-WALK-2 wins in noisy conditions.