Annals of Mathematics and Artificial Intelligence 24 (1998) 211–223

Efficiently searching a graph by a smell-oriented vertex process *

Israel A. Wagner [a,b], Michael Lindenbaum [b] and Alfred M. Bruckstein [b]

[a] *IBM Haifa Research Lab, Matam, Haifa 31905, Israel*
E-mail: israelw@vnet.ibm.com
[b] *Department of Computer Science, Technion City, Haifa 32000, Israel*
E-mail: {mic;freddy}@cs.technion.ac.il

Efficient graph search is a central issue in many aspects of AI. In most of existing work there is a distinction between the active "searcher", which both executes the algorithm and holds the memory, and the passive "searched graph", over which the searcher has no control at all. Large dynamic networks like the Internet, where the nodes are powerful computers and the links have narrow bandwidth and are heavily-loaded, call for a different paradigm, in which most of the burden of computing and memorizing is moved from the searching agent to the nodes of the network. In this paper we suggest a method for searching an undirected, connected graph using the Vertex-Ant-Walk method, where an a(ge)nt walks along the edges of a graph G, occasionally leaving "pheromone" traces at nodes, and using those traces to guide its exploration. We show that the ant can cover the graph within time O(nd), where n is the number of vertices and d the diameter of G. The use of traces achieves a trade-off between random and self-avoiding walks, as it dictates a lower priority for already-visited neighbors. Further properties of the suggested method are: (a) modularity: a group of searching agents, each applying the same protocol, can cooperate on a mission of covering a graph with minimal explicit communication between them; (b) possible convergence to a limit cycle: a Hamiltonian path in G (if one exists) is a possible limit cycle of the process.

1. Introduction

Following an ancient advice,[1] we consider a memoryless a(ge)nt that searches a graph G for food. The ant has the ability to leave pheromone traces on vertices, and to sense the smell traces at the current location and its immediate neighbors. By "sensing the smell" at a vertex we mean that the ant knows both the number of smell traces that have been left on the vertex, and the time of the most recent trace. Formally, a vertex v at time t is marked by a pair $(\sigma_t(v), \tau_t(v))$, where $\sigma_t(v)$ is the number of marks left on v, and $\tau_t(v)$ is the time of the most recent mark left there up to time t. Initially we set both marks to zero for all $v \in V$, V being the set of vertices. Being at a vertex u at time t, the ant smells around and chooses among $N(u)$, the

* An earlier version of this paper was presented in AAAI '97 Workshop on On-Line Search.
[1] "Go to the ant, thou sluggard; consider her ways, and be wise" (Proverbs, vi, 6).

set of neighbors[2] of u, a neighbor v with the minimum mark on it, where marks are ordered lexicographically by the (σ, τ) values on the vertices. Assuming the minimum was found on neighbor v, the ant then sets $\sigma(u)$ to $\sigma(v) + 1$, and $\tau(u)$ to the current time. Then it goes to v. Intuitively, this rule of motion behaves like a steepest-decent optimizer, with the additional option to dynamically alter the cost function, thereby avoiding being stuck in a local minima like steepest-decent methods.

We shall call this process "Vertex-Ant-Walk" (VAW for short) to distinguish it from a similar "Edge-Ant-Walk" process discussed in [19,20] where the edges, rather than the vertices, were marked. It has been shown there that a group of k smell-oriented ants that evolve in an edge process can cover a graph in a (worst case) time of $O(\Delta n^2/k)$, where Δ is the maximum vertex-degree and n the number of vertices. The method suggested in the current paper covers a connected graph using no more than nd steps, where d is the diameter[3] of G. This is an improvement since, in general, $nd < \Delta n^2$.

The VAW process, beyond its theoretical interest, has applications in robotics where a robot with limited sensing capabilities but with the ability to leave marks on the ground has to cover a closed region for purposes of cleaning a dirty floor, painting a wall, or demining a mine-field. Another potential application is searching a large network of WWW sites which are changing slowly; if a web-site does not change more than once in T units of time, and $T < dn$, then our method guarantees that no change will be missed.

Related work: Graph search is an old problem; several methods exist for deterministic (e.g., [7,8,11,15,16]) random (e.g., [1,2,5]) and semi-random [9] covering. A step towards a trace-oriented theory of search was done in [3,4], where *pebbles* are used to assist the search. Pebbles are tokens that can be placed on the floor and later be removed. In [3] it was shown that a finite automaton with two pebbles can search all mazes. 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, by letting them lose value with time. There are several related search methods in the literature, some of which will be briefly mentioned in the sequel. The RTA* (*Real-Time A**) and LRTA* (*Learning RTA**) [12] are two variations on the famous A* heuristic search, with a cost function that takes the current searcher's location into account, thus making the algorithm more realistic for field applications like robotics. In [17,18] a *counter-based exploration* method is described which is quite similar to our method, but the upper bound shown there on the cover time is $O(n^2 d)$, where d is the diameter and n is the number of vertices in the graph. In the *Nearest Neighbor Approach* (NNA) method of [13], a graph is learned by moving towards the nearest un-visited edge in the graph, and an upper bound is proved on the cover time of $O(m \log n)$ where m is the number of edges in the graph. The usage of ideas from nature for search and optimization problems has recently acquired popularity. See [6] and references therein for an ant-system

[2] $N(u)$, the set of neighbors of vertex u in a graph $G(V, E)$, is the set $\{v \mid u \neq v, (u, v) \in E\}$.

[3] The diameter of a graph $G(V, E)$, denoted diam(G), is defined as the maximum distance between any pair of vertices in the graph, i.e., $\text{diam}(G) = \max_{u,v \in V}\{\text{length of the shortest } (u\text{–}v) \text{ path}\}$.

used to solve *Travelling Salesperson* (TSP) problems. A different way to achieve cooperation by changing the common environment was presented in [10], where state-changing rules are applied by the agents in order to jointly solve a Tileworld problem.

Our algorithm suggests a reasonable trade-off between the rigid, highly sensitive *Depth-First Search* (DFS) and self avoiding [14] walk on one hand, and the absolutely adaptive (but very time-consuming) random walk, on the other hand.

In this paper we prove an O(nd) bound on the *cover time* of a graph by the above process, where $n = |V(G)|$ and $d = \text{diam}(G)$, and show that a Hamiltonian cycle, if one exists in G, is a possible limit cycle of the process.

2. Vertex-Ant-Walk – a tradeoff between random and self-avoiding walks

Formally, the process is defined by the following rule of motion:

Rule Vertex-Ant-Walk(**u** vertex)
(A) $v := u'$s neighbor with minimal value of $(\sigma(\cdot), \tau(\cdot))$;
(if there is a tie – make a random decision)
(B) $\sigma(u) := \sigma(v) + 1$;
(D) $\tau(u) := t$;
(E) $t := t + 1$;
(F) go to v.
end Vertex-Ant-Walk.

Initially all (σ, τ) values are set to $(0, 0)$ and a starting vertex is chosen, and then the rule is applied repeatedly. The tie referred to in step A can only occur between yet unvisited vertices; once a vertex is visited, it always have a unique time stamp that avoids further ties. In the sequel we shall refer to the above rule as the VAW rule. See figures 1 and 7 for examples of the VAW evolution.

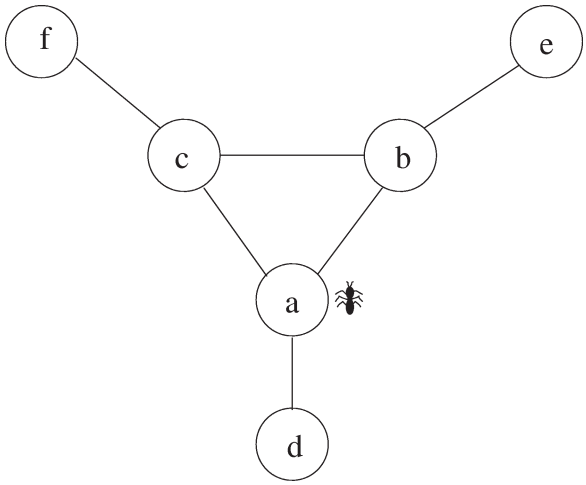
Note that more than one ant may occupy a vertex at the same time; however the ants are exiting the vertex at slightly different times. The order of exiting the vertex can be determined by some priority between the ants, which may be determined by assigning a unique hard-coded ID to each ant. Another way to resolve such conflicts may be to cause each ant to have a slightly different phase on its clock, or even use a random phase, which should avoid phase-collisions with high probability.

How efficient is this process? Let us now show that a single ant covers any connected graph within O(nd) steps, where n is the number of vertices and d is the diameter of the graph.

Lemma 1. If (u, v) is an edge in G then it always holds that

$$\big|\sigma(u) - \sigma(v)\big| \leqslant 1.$$

Proof. The lemma is clearly true when $t = 0$ since all $\sigma(\cdot)$ values are being preset to zero. Assuming it is also true at time t, we claim that the $(t + 1)$th step of VAW



(a,1,1)	(b,1,2)	(c,1,3)	(f,2,4)	(c,2,5)	(a,1,6)
(d,2,7)	(a,2,8)	(b,1,9)	(e,2,10)	(b,3,11)	(c,3,12)
(f,4,13)	(c,3,14)	(a,3,15)	(d,4,16)	(a,4,17)	(b,3,18)
⋮					⋮

Figure 1. The first 18 steps in a VAW tour of a graph with 6 vertices and diameter 3, starting from vertex a. The corresponding $(u_t, \sigma(u_t), \tau(u_t))$ evolution is shown in the table. The process starts at time $t = 1$, and $\sigma(u_t)$ is the value after the tth step has been completed. Note that the graph is covered in the 10th step (upon first visiting vertex e), and the sequence of vertices in the tour is the regular expression $ab\{cfcadabeb\}^*$ (i.e., the limit cycle is of length 9).

does not cause any harm, i.e., the assertion of the lemma remains true. Note that the only change in σ at this time may occur at the current node u in step B of the VAW rule. Hence we only need to show that even after the $(t + 1)$th step, all differences $|\sigma(u) - \sigma(v)|$ remain less than or equal to 1. But since $\sigma(v)$ is the minimum among u's neighbors, the maximum difference after step B is between u and v; this difference, however, is equal to 1, so the assertion holds. □

Our next step is to show that under the VAW rule, the sum of all σ-markers at the end of the tth step is at least t. For this end, let us define the total smell on the graph at time t, $\sigma_t(G)$, to be the sum of all the σ values on the vertices at that time.

Lemma 2. At all times t,

$$\sigma_t(G) \stackrel{\triangle}{=} \sum_{u \in V} \sigma_t(u) \geqslant t.$$

Proof. Let us denote the sequence of nodes visited up to time t by $u_1, u_2, \ldots, u_t$, and the σ-value of a vertex u upon completion of time-step t, by $\sigma_t(u)$. The only change

to σ may occur at the vertex currently visited; hence

$$\sigma_t(G) = \sigma_{t-1}(G) + \delta_t,$$

where δ_i stands for the addition to $\sigma(G)$ at time i, i.e.,

$$\delta_i = \sigma_i(u_i) - \sigma_{i-1}(u_i). \tag{1}$$

According to the VAW rule, upon moving from u_i to u_{i+1}, the value of $\sigma_i(u_i)$ is set to $\sigma_i(u_{i+1}) + 1$; hence

$$\delta_i = \sigma_i(u_{i+1}) - \sigma_{i-1}(u_i) + 1. \tag{2}$$

From equation (1),

$$\sigma_t(u_t) = \sigma_{t-1}(u_t) + \delta_t, \tag{3}$$

and, substituting $i = t - 1$ in equation (2),

$$\sigma_{t-1}(u_t) = \sigma_{t-2}(u_{t-1}) + \delta_{t-1} - 1. \tag{4}$$

Applying equations (3) and (4) recursively one gets

$$\begin{aligned}
\sigma_t(u_t) &= \sigma_{t-1}(u_t) + \delta_t \\
&= \sigma_{t-2}(u_{t-1}) + \delta_{t-1} + \delta_t - 1 \\
&= \sigma_{t-3}(u_{t-2}) + \delta_{t-2} + \delta_{t-1} + \delta_t - 2 \\
&\vdots \\
&= \sigma_0(u_1) + \sum_{i=1}^{t} \delta_t - (t-1),
\end{aligned}$$

and hence, since $\sigma_t(G) = \sum_{i=1}^{t} \delta_i$, we get that

$$\sigma_t(G) = \sigma_t(u_t) - \sigma_0(u_1) + t - 1.$$

But, according to our rule, $\sigma_0(u_1) = 0$ and $\sigma_t(u_t) \geqslant 1$, and the lemma follows. □

Now let us combine the smoothness of $\sigma(\cdot)$ (lemma 1) together with its temporal accumulation (lemma 2) to get

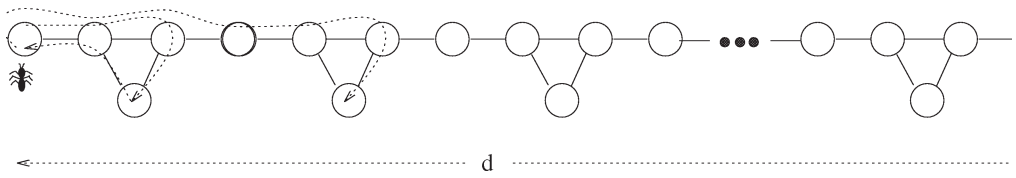


Figure 2. A hard case for the VAW rule. There are n vertices, and about $1.25n$ edges. The diameter is about $0.8n$, and the time needed to traverse it may be as long as $O(nd) = O(n^2)$. The dotted arrows show the worst case where each triangle of vertices is a "trap" that causes the ant to go back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

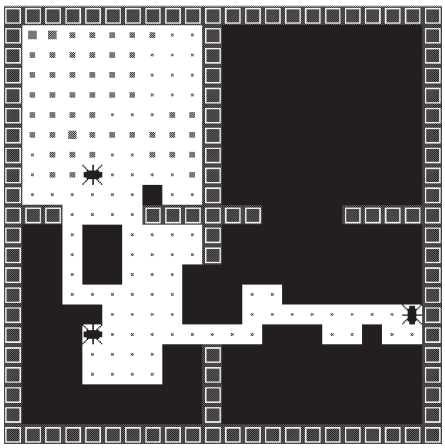


Figure 3. A simulation of the VAW algorithm with three ants. A visited cell u is marked by a white square, and the size of the gray square within a square u is proportional to $\sigma(u)$ – the level of trace on u. Black squares have not yet been visited.

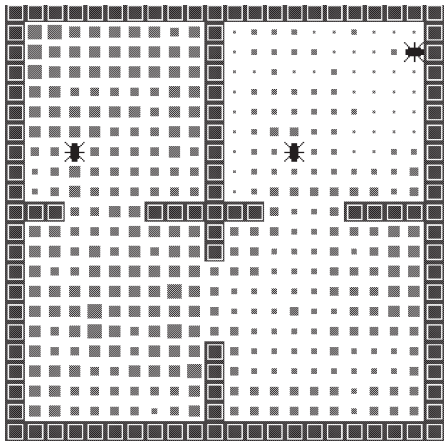


Figure 4. Upon completion, the whole graph is covered. This happened after 362 time-steps in this 20×20 arena.

Theorem 1. Denote by d the diameter of G, and by n the number of vertices. Then after at most nd steps the graph G is covered.

Proof. We shall prove a slightly stronger claim: for all $u \in V$, $\sigma_{d(2n-d-1)/2}(u) > 0$. In other words, after at most $d(2n-d-1)/2$ steps the graph is covered. Assume that at time t the graph has not yet been fully covered. Then there exists at least one node, say u, for which $\sigma(u) = 0$. According to lemma 1, none of u's neighbors can have $\sigma > 1$, none of the neighbors' neighbors can have $\sigma > 2$, etc. Hence we have at least one node with $\sigma = 0$, at least one with $\sigma = 1$, and so on up to $d-1$. The rest of the nodes (at most $n-d$) have $\sigma \leqslant d$. This gives a total $\sigma(G)$ of at most $0+1+2+\cdots+d-1+d(n-d) = d(d-1)/2+d(n-d) = d(2n-d-1)/2$. But according to lemma 2, the total value of σ at time t is at least t; hence if $t > d(2n-d-1)/2$ the graph is necessarily covered. □

The bound of O(nd) is tight; the example in figure 2 depicts a case where the cover time required by VAW is indeed O(nd).

3. Simulation of the VAW process

A simulator of the process for one or several ants has been written in Java[4] and placed on the web. Since in our paradigm the memory (i.e., the set of (σ, τ) values)

[4] The simulator is web-accessible through: `http://www.cs.technion.ac.il/~wagner/pub/vaw.html`.

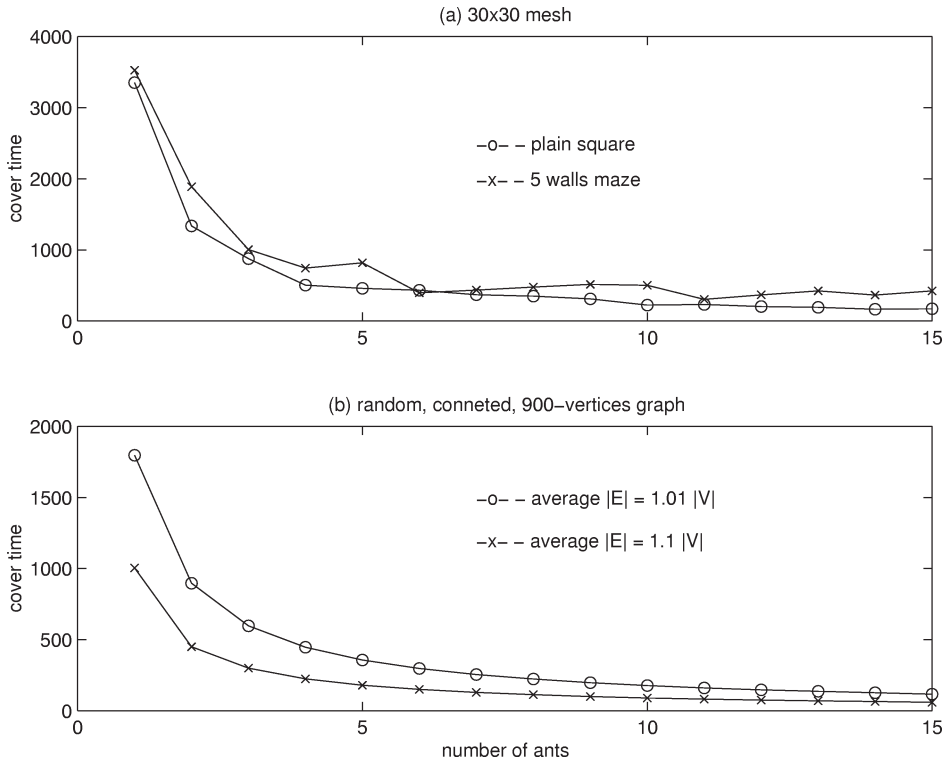


Figure 5. Times of complete coverage by the VAW process, plotted against the number of ants. Figure (a) shows the results for a 30×30 mesh, with and without maze-walls, while in figure (b) a connected random graph was drawn by adding random edges to a 900-vertices tree. The experiment was conducted 20 times and the average cover times are plotted, for average edge numbers of $1.01n$ and $1.1n$, showing a faster covering (on the average) as density grows (but see also the next figure). As the number of ants increases, it sometimes happens that one disturbs the others; this explains the slight non-monotonicity of some of the plots.

is distributed among the nodes, several ants can work together in covering a graph without any explicit communication. See figures 3, 4 for an example of a simulation on a planar mesh graph.

Simulation results show that, as long as k, the number of ants, is small relative to the number of nodes, the speed-up ratio (defined as T_k/T_1) comes close to $1/k$. See figure 5 for cover times vs. number of agents, and figure 6 for cover time vs. average edge density in random graphs. Also clear from simulations is the fact that in most cases, covering is much faster than the upper bound of nd.

4. VAW and the Hamilton cycle

As opposed to DFS and similar search methods, our VAW ant never stops, unless we pre-program it with an upper bound on its walking time. Hence, VAW is not

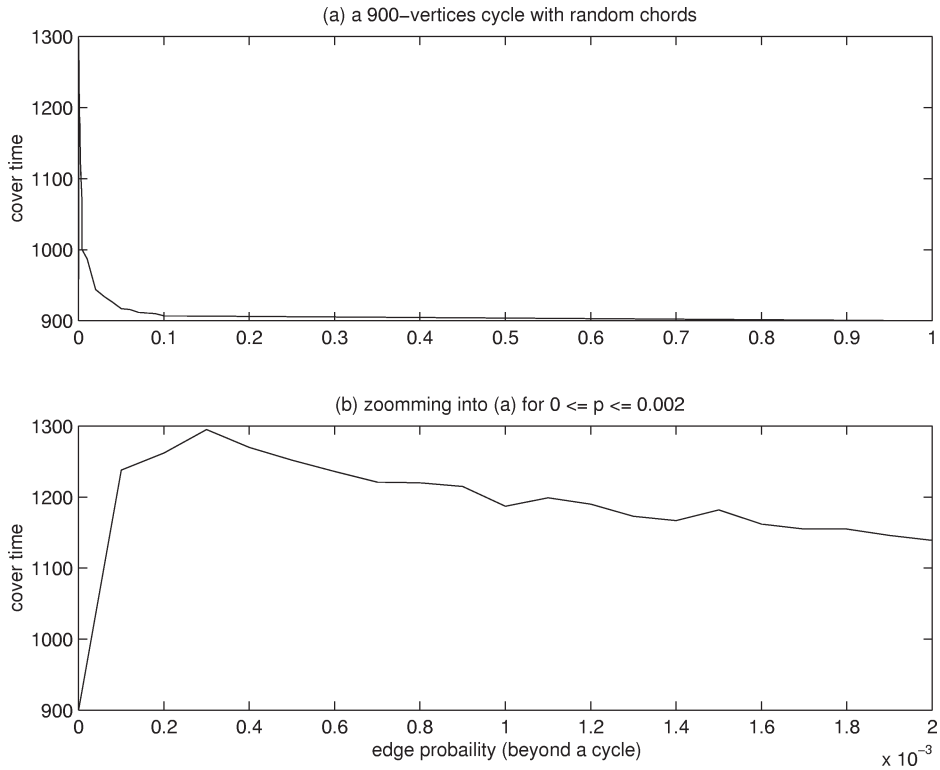
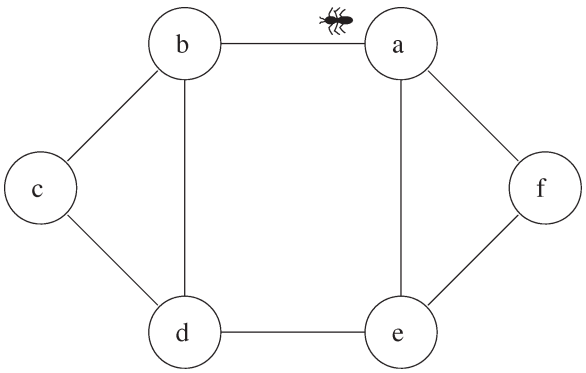


Figure 6. Times of complete coverage by the VAW process on random graphs generated by adding random chords to a 900-vertices cycle, where each possible chord (there are 403650) is added with probability p. Average cover times are plotted against the edge probability, first (a) for $0 \leqslant p \leqslant 1$, and then (b) for $0 \leqslant p \leqslant 0.002$. The peak at $p = 0.0003$ may be explained by the higher probability for traps (as in figure 2) in lower densities.

the best method for a one-shot search, when compared to those methods which are aware of their status of completion. However, as we shall show in this section, it may be a competitive candidate for an adaptive-repetitive search, in which we want the graph to be searched again and again. Applications may be guarding/surveillance or keeping an up-to-date database in the context of a huge network of data sources (e.g., Internet).

Let us assume that after covering the graph we allow our ant to continue its VAW process. It follows from theorem 1 that once every (at most) dn steps, another coverage of the graph is completed. If the graph has a Hamiltonian cycle,[5] following this cycle is clearly the shortest way to cover the graph. See figure 7 for an example of a Hamiltonian cycle discovered by the process. We shall now show that a Hamiltonian cycle (if one exists in G) is a possible limit cycle of the VAW process. More

[5] A closed path in a graph is a *Hamiltonian cycle* if it visits each vertex of the graph exactly once, with the only exception of the first and last vertex.



(a,1,1)	(b,1,2)	(d,1,3)	(c,2,4)	(b,2,5)	(a,1,6)
(f,1,7)	(e,2,8)	(d,3,9)	(c,3,10)	(b,2,11)	(a,2,12)
(f,3,13)	(e,3,14)	(a,3,15)	(b,4,16)	(d,4,17)	(c,5,18)
(b,4,19)	(a,4,20)	(f,4,21)	(e,5,22)	(d,5,23)	(b,5,24)
(a,5,25)	(f,6,26)	(e,6,27)	(d,6,28)	(c,6,29)	(b,6,30)
(a,7,31)	(f,7,32)	(e,7,33)	(d,7,34)	(c,7,35)	(b,8,36)
⋮					⋮

Figure 7. A VAW ant covers a graph, starting from vertex a, and arrives at a Hamiltonian limit cycle at time 24. The sequence of vertices is: *abdcbafedcbafeabdcbafed*{*bafedc*}*.

specifically, if the ant happens to follow such a cycle for n consecutive times (i.e., n^2 steps), it will follow this cycle forever.

Lemma 3. A Hamiltonian cycle in G, upon being traversed n consecutive times, becomes a limit cycle of the VAW process.

Proof. Recall that according to the VAW rule, upon leaving a vertex, its σ-value is set to the current value of the next vertex plus one, i.e.,

$$\sigma_t(u_t) = \sigma_t(u_{t+1}) + 1. \tag{5}$$

Let us now assume that at time t the ant has completed a sequence of n Hamiltonian cycles. Then u_{t+1} has not been visited (and its σ-value has not been changed) since time $t - n + 1$ and hence

$$\sigma_t(u_{t+1}) = \sigma_{t-n+1}(u_{t+1}) = \sigma_{t-n+1}(u_{t-n+1}), \tag{6}$$

the last equality being due to the fact that the vertex indices are all mudulo n, since, in our case, the same sequence of n vertices is repeated n times. One can now substitute equation (6) into equation (5) to get

$$\begin{aligned}\sigma_t(u_t) &= \sigma_{t-n+1}(u_{t-n+1}) + 1 \\ &= \sigma_{t-2(n+1)}(u_{t-2(n+1)}) + 2\end{aligned}$$

$$
\begin{aligned}
&= \sigma_{t-3(n+1)}(u_{t-3(n+1)}) + 3 \\
&\vdots \\
&= \sigma_{t-n(n+1)}(u_{t-n(n+1)}) + n \\
&= \sigma_{t-n(n+1)}(u_t) + n.
\end{aligned}
$$

A conclusion from the last equation is that during the $(n+1)$st cycle, the ant sees (or smells) the same relative σ-pattern that it saw in the first cycle; the n added to

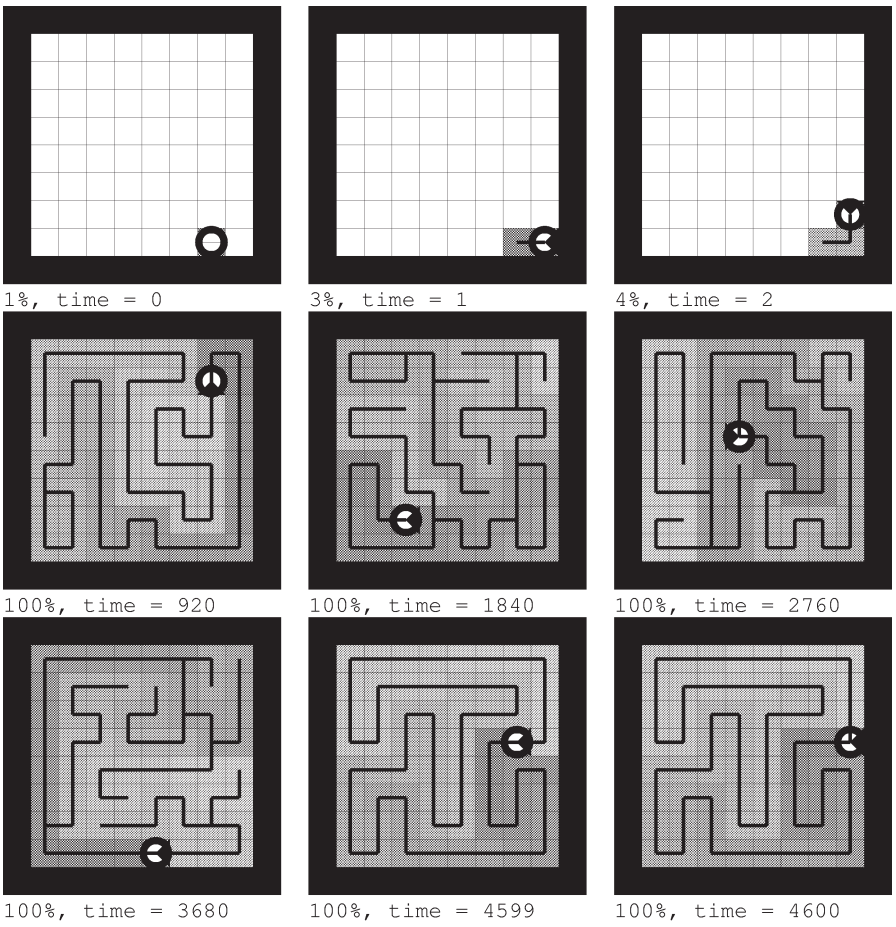
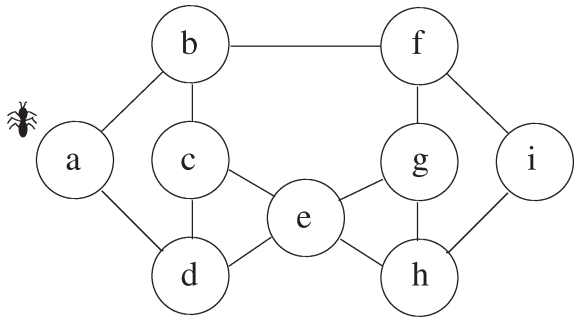


Figure 8. The VERTEX-ANT-WALK dynamics has the interesting property that cycle covers of the graph are among the limit cycles. Here a Hamiltonian path (i.e., a special case of cycle-cover) was found as a limit cycle of the process. The black lines describe the edges traversed by the ant in the most recent n units of time.



(a,1,1)	(b,1,2)	(c,1,3)	(e,1,4)	(h,1,5)	(i,1,6)
(f,1,7)	(g,1,8)	(e,1,9)	(d,2,10)	(a,2,11)	(b,2,12)
(c,2,13)	(e,2,14)	(h,2,15)	(i,2,16)	(f,2,17)	(g,3,18)
(e,3,19)	(d,3,20)	(a,3,21)	(b,3,22)	(c,4,23)	(e,3,24)
⋮					⋮

Figure 9. A Hamiltonian 9-vertex graph which is not recognized by the VAW process. The sequence of traversed vertices in this case is $\{abcehifged\}^*$, while the Hamiltonian cycle is *abfihgecd*.

each value does not change the "greater than" relations between the vertices. Also, the τ-pattern is obviously equivalent to the previous one; hence, the same decisions as to "where to go from here" should be taken again. The same assertion also holds for the $(n+2)$nd cycle, the $(n+3)$rd cycle, and so on. □

See figure 8 for an example of the the Hamiltonian cycle generated by the VAW process. Note, however, that a Hamiltonian cycle is not *always* detected by a VAW process, as shown in figure 9.

5. Some open questions

The vertex ant walk seems to be quite a simple process; however, several facets of its behavior are still challenging, as is shown by the following examples.

- Our cover time result holds for a single ant. It is not in general true that adding ants reduces cover time, as can be seen from the plots in figure 5. However it seems reasonable to say that if the graph is large and the ants are fairly distributed in the beginning, the load will be more or less balanced. Is it possible to quantify the speed-up ratio (i.e., T_k/T_1) for k ants working together, as a function of k and some parameters of the graph?
- We have shown (lemma 3) that if a Hamiltonian cycle is traveresed n times it becomes a limit cycle. However we have not found an example in which a Hamiltonian cycle, once traversed, does *not* become a limit cycle. So one may ask if we really need the ant to follow the cycle n times, or can a smaller number of repetitions be sufficient to assure a limit cycle.

- Under which conditions does a VAW process on a Hamiltonian graph converge to a limit cycle? More specifically, denote by R_n the set of n-vertex Hamiltonian graphs which are recognized by a VAW process , and by H_n the set of all n-vertex Hamiltonian graphs. Can the ratio R_n/H_n be estimated? One may also ask, for a given graph in $H_n \setminus R_n$ (like the graph in figure 9) , what is the probability of a VAW process to find the Hamiltonian cycle, taking into account the fact that the first several steps are not uniquely determined by the rule – it is only after the graph is covered and each vertex has a unique τ value that the steps are unique.
  A related but different question is about general graphs: does the spanning tree induced by VAW (i.e., the tree spanned by the edges that connect each vertex to its neighbor with highest τ-value) improve (in terms of covering duration) with time? Simulations are positive but we seek an analytical argument.
- A probabilistic version of VAW rule does not determine the next neighbor specifically, but assigns each neighbor a probability according to its current (σ, τ) mark, e.g., the probability of jumping from u to v may be

$$\mathsf{Prob}(u \to v) = \frac{1/(1+\sigma(v))}{\sum_{w \in N(u)} 1/(1+\sigma(w))},$$

  where, clearly, $\sum_{w \in N(u)} \mathsf{Prob}(u \to w) = 1$. Is such a trace-biased random coverage process faster, on the average, or slower than the deterministic one?

Acknowledgements

We thank Bob Holt of AT&T–Bell Labs for his comments on an early version of this paper, and Uri Feige of the Weizmann Institute for the counterexample in figure 9. We also wish to thank the anonymous referees for their careful reading and helpful comments.

References

[1] R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovasz and C. Rakoff, Random walks, universal traversal sequences, and the complexity of maze problems, in: *20th Annual Symposium on Foundations of Computer Science*, San Juan, Puerto Rico (October 1979) pp. 218–223.
[2] G. Barnes and U. Feige, Short random walks on graphs, in: *Proc. of the 25th ACM STOC* (1993).
[3] M. Blum and D. Kozen, On the power of the compass, or, Why mazes are easier to search than graphs, in: *Proc. of FOCS '78* (1978) pp. 132–142.
[4] M. Blum and W.J. Sakoda, On the capability of finite automata in 2- and 3-dimensional space, in: *Proc. of FOCS '77* (1977) pp. 147–161.
[5] A.Z. Broder, A.R. Karlin, P. Raghavan and E. Upfal, Trading space for time in undirected s–t connectivity, SIAM J. Comput. 23(2) (1994) 324–334.
[6] M. Dorigo, V. Maniezzo and A. Colorni, The ant system: Optimization by a colony of cooperating agents, IEEE Trans. on Systems, Man, and Cybernetics – Part B 26 (1996) 29–41.
[7] S. Even, *Graph Algorithms* (Computer Science Press, Rockville, MD, 1979).

[8] A.S. Fraenkel, Economic traversal of labyrinths, Math. Mag. 43 (1970) 125–130, and a correction in 44 (1971) 12.
[9] S. Gal and E.J. Anderson, Search in a maze, Probab. Engrg. Inform. Sci. 4 (1990) 311–318.
[10] C.V. Goldman and J.S. Rosenschein, Emergent coordination through the use of cooperative state-changing rules, in: *Proc. of The National Conference on AI*, Seattle, WA (August 1994) pp. 408–413.
[11] J. Hopcroft and R. Tarjan, Efficient algorithms for graph manipulation, Comm. ACM (1973) 372–378.
[12] R.E. Korf, Real-time heuristic search, Artificial Intelligence 42 (1990) 189–211.
[13] S. Koenig and Y. Smirnov, Graph learning with a nearest neighbor approach, in: *Proc. of COLT '96*, Desenzano del Garda, Italy (June 28–July 1, 1996).
[14] N. Madras and G. Slade, *The Self-Avoiding Walk* (Birkhäuser, Basel, 1993).
[15] R. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1(2) (1972) 146–160.
[16] G. Tarry, Le problem des labyrinths, Nouvelles Annales de Mathematiques 14 (1895) 187.
[17] S. Thrun, The role of exploration in learning control, in: *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches* (Van Nostrand Reinhold, Florence, KY, 1992).
[18] S. Thrun, Efficient Exploration in Reinforcement Learning, Technical Report CMU-CS-92-102, Carnegie-Mellon Univ., Pittsburgh, PA (1992).
[19] I.A. Wagner, M. Lindenbaum and A.M. Bruckstein, Smell as a computational resource – a lesson we can learn from the ant, in: *Proc. of ISTCS '96* (1996) pp. 219–230. Web accessible through: `http://www.cs.technion.ac.il/~wagner`.
[20] I.A. Wagner, M. Lindenbaum and A.M. Bruckstein, Cooperative covering by ant-robots using evaporating traces, Technical Report CIS-9610, Center for Intelligent Systems, Technion, Haifa (April 1996); to appear in IEEE Trans. Robot. Autom.