Efficient and Inefficient Ant Coverage Methods

Sven Koenig [a] Boleslaw Szymanski [b] Yaxin Liu [c]

[a] *College of Computing, Georgia Institute of Technology*
E-mail: skoenig@cc.gatech.edu
[b] *Department of Computer Science, Rensselaer Polytechnic Institute*
E-mail: szymansk@cs.rpi.edu
[c] *College of Computing, Georgia Institute of Technology*
E-mail: yxliu@cc.gatech.edu

Ant robots are simple creatures with limited sensing and computational capabilities. They have the advantage that they are easy to program and cheap to build. This makes it feasible to deploy groups of ant robots and take advantage of the resulting fault tolerance and parallelism. We study, both theoretically and in simulation, the behavior of ant robots for one-time or repeated coverage of terrain, as required for lawn mowing, mine sweeping, and surveillance. Ant robots cannot use conventional planning methods due to their limited sensing and computational capabilities. To overcome these limitations, we study navigation methods that are based on real-time (heuristic) search and leave markings in the terrain, similar to what real ants do. These markings can be sensed by all ant robots and allow them to cover terrain even if they do not communicate with each other except via the markings, do not have any kind of memory, do not know the terrain, cannot maintain maps of the terrain, nor plan complete paths. The ant robots do not even need to be localized, which completely eliminates solving difficult and time-consuming localization problems. We study two simple real-time search methods that differ only in how the markings are updated. We show experimentally that both real-time search methods robustly cover terrain even if the ant robots are moved without realizing this (say, by people running into them), some ant robots fail, and some markings get destroyed. Both real-time search methods are algorithmically similar, and our experimental results indicate that their cover time is similar in some terrains. Our analysis is therefore surprising. We show that the cover time of ant robots that use one of the real-time search methods is guaranteed to be polynomial in the number of locations, whereas the cover time of ant robots that use the other real-time search method can be exponential in (the square root of) the number of locations even in simple terrains that correspond to (planar) undirected trees.

Keywords: Ant Robots, Cover Time, Graph Coverage, Lawn Mowing, Learning Real-Time A*, Node Counting, Real-Time Heuristic Search, Surveillance, Vacuum Cleaning

Ants (that is, ant robots) are simple creatures with limited sensing and computational capabilities. Ants have the advantage that they are simple to design,

easy to program, and cheap to build. This makes it feasible to deploy groups of ants and take advantage of the resulting fault tolerance and parallelism [6]. We study the behavior of ants for covering terrain, as required for vacuum cleaning, lawn mowing, mine sweeping, and surveillance. Vacuum cleaning and lawn mowing, for example, are realistic applications for ants given that the first cheap and small vacuum-cleaning and lawn mowing household robots are already on the consumer market or are expected to be on the consumer market soon, including the Koala robot, DC06 robot, Electrolux robot, Cye robot, Eureka robot, Robomow robot, Solar Mower robot, and Dophin robot. We study both one-time coverage and continuous coverage, where ants continuously cover a large terrain without getting switched off, so that every part of the terrain gets visited once every while. Ants cannot use conventional planning methods due to their limited sensing and computational capabilities which limit their planning capabilities even for simple planning tasks such as path planning or the coverage of terrain. For example, ants might not be able to maintain maps and use them for path planning. In fact, they might not be able to plan complete paths at all. Thus, they might not be able to cover terrain as efficiently as robots with more powerful sensing and computational capabilities. On the other hand, groups of ants can take advantage of both their fault tolerance (they fail gracefully even if some ants malfunction) and their parallelism (groups of ants can cover terrain faster than a single ant).

We study ants that use real-time search methods. Real-time (heuristic) search methods [17] interleave planning and plan execution. They can be used by single ants as well as groups of ants, and require the ants to have only very limited computational capabilities and look-aheads. The ants only have to leave markings in the terrain, sense markings at their neighboring locations, and change the marking of their current location. These markings can be sensed by all ants and allow them to cover terrain even if they do not have any kind of memory, cannot maintain maps of the terrain, nor plan complete paths. This is what some insects do [1]. Unfortunately, not much is known about the properties of real-time search methods since the area of real-time search is mostly an experimental one and not many theoretical results are known. Real-time search methods differ in this aspect from traditional search methods, whose properties have been researched extensively. We therefore study, both theoretically and experimentally, the behavior of ants that use two simple real-time search methods to cover terrain. We show experimentally that both real-time search methods robustly cover terrain even if the ant robots are moved without realizing this (say, by people running into them), some ant robots fail, and some markings get destroyed. The two real-time search methods differ only in how they update the markings. They are algorithmically similar, and experimental results indicate that the cover time of ants that use them is often similar on grids. Our analysis is therefore surprising. We show that the cover time of ants that use one of the real-time search methods is guaranteed to be polynomial in the number of locations, whereas the

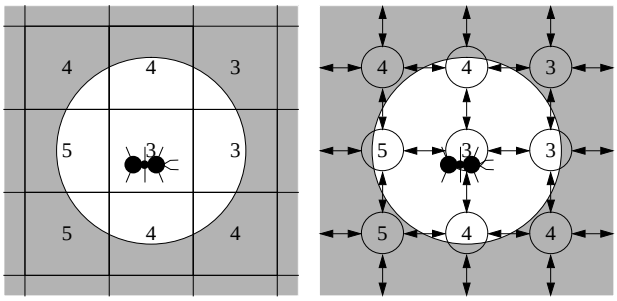


Figure 1. Ants and Real-Time Search

cover time of ants that use the other real-time search method can be exponential in (the square root of) the number of locations even in simple terrains that correspond to (planar) undirected trees. These results demonstrate that experimental comparisons of the behavior of ants that use real-time search methods are often insufficient to evaluate how well they scale up because the cover time of ants that use two similar real-time search methods can be very different even if experimental results seem to indicate otherwise. An analysis can help to detect these problems and prevent surprises later on, as well as provide a solid theoretical foundation for ant coverage with real-time search methods.

1. Task and Notation

Terrains can be modeled as (directed) graphs with their locations corresponding to the vertices, for example by imposing regular grids on them. Figure 1 (left), for example, shows a regular four-connected grid, and Figure 1 (right) shows the corresponding graph. We do not impose restrictions on the topology of the graphs (unless stated otherwise), except that we require them to be strongly connected (every vertex can be reached from every other vertex). Strongly connected graphs guarantee that ants are always able to reach all vertices, no matter which edges they have traversed in the past. For simplicity, we assume that several ants can be at the same vertex at the same time. Their task is to cover the graph (visit all vertices) once or repeatedly. We use the following notation in this article: S denotes the finite set of vertices (states) of the graph, and $s_{start} \in S$ denotes the start vertex. The number of vertices is $n = |S|$. $A(s) \neq \emptyset$ is the finite, nonempty set of (directed) edges leaving vertex $s \in S$ (actions that can be executed in state s). $succ(s, a)$ denotes the successor vertex that results from the traversal of edge $a \in A(s)$ in vertex $s \in S$. We say that a graph is undirected iff, for every directed edge between two vertices, there is also a corresponding directed edge between the vertices in the opposite direction. To simplify matters, we measure the distances in edge traversals throughout this article, which is justified if the costs of all edges are roughly the same. We also use two operators with the following semantics: Given a finite set X, the expression "one-of X"

Initially, the u-values $u(s)$ are zero for all $s \in S$.

1. $s := s_{start}$.
2. $a :=$ one-of $\arg\min_{a \in A(s)} u(succ(s,a))$.
3. Update $u(s)$ using the value-update rule.
4. Traverse edge a.
5. $s := succ(s,a)$.
6. Go to 2.

Figure 2. Real-Time Search with Look-Ahead One

returns an element of X according to an arbitrary rule. A subsequent invocation of "one-of X" can return the same or a different element. The expression "$\arg\min_{x \in X} f(x)$" returns the elements $x \in X$ that minimize $f(x)$, that is, the set $\{x \in X | f(x) = \min_{x' \in X} f(x')\}$, where f is a function from X to the non-negative integers.

2. Real-Time Search

Real-time (heuristic) search methods [16] interleave planning and plan execution, and allow for fine-grained control over how much planning to perform between plan executions. They are related to reinforcement-learning methods [10,25], as pointed out in [3]. Planning is done via local searches, that is, searches that are restricted to the part of the graph around the current vertex of an agent. The white area of Figure 1 illustrates the limited look-ahead of real-time search methods. Experimental evidence suggests that real-time search methods are efficient domain-independent search methods that outperform traditional search methods such as A* [18] on a variety of search problems. Real-time search methods have, for example, successfully been applied to traditional search problems [17], moving-target search problems [9], STRIPS-type planning problems [5], robot navigation and localization problems with initial pose uncertainty [14], totally observable Markov decision process problems [3], and partially observable Markov decision process problems [4], among others. A good overview of real-time search methods is given in [8] and newer developments can be found in [11]. We study two similar real-time search methods that ants can use to cover graphs, namely variants of Learning Real-Time A* and Node Counting that have a look-ahead of only one edge traversal and fit the algorithmic skeleton shown in Figure 2. Both real-time search methods associate a u-value $u(s)$ with each vertex $s \in S$ and initialize them with zeroes. The semantics of the u-values depend on the real-time search method but are simple and result in intuitive rules of thumb for which edges to traverse. The u-values correspond to markings that the ants leave at the vertices of the graphs, see Figure 1. This is what some insects do. Real ants, for example use chemical (pheromone) traces to guide their navigation [1]. Mobile robots have been built that leave markings in the terrain

but so far only short-lived markings such as odor traces [22], heat traces [21], or alcohol traces [24]. One challenge is to build mobile robots that leave long-lived markings in the terrain, especially long-lived markings that do not get destroyed by activities such as vacuum cleaning or lawn mowing.

Ants that use real-time search methods do not need to know or learn the graph (and can thus operate on known and unknown graphs) nor compute complete paths. Rather, they always decide which neighboring vertex to move to based only on the u-values of the neighboring vertices. Before moving to that vertex, the ants can change the u-value of their current vertex. Both Learning Real-Time A* and Node Counting first decide which edge the ants should traverse in their current vertex (Line 2). They look one edge traversal ahead (larger look-aheads are possible) and always greedily let the ants traverse the edge that leads to a neighboring vertex with a smallest u-value (ties can be broken arbitrarily). Then, they update the u-value of their current vertex using a value-update rule that depends on the semantics of the u-values and thus the real-time search method (Line 3). We assume that the ants execute Lines 2 and 3 together and atomically (for example, because the body of the ant covers the current location and its neighbors). Finally, both real-time search methods let the ants traverse the selected edge (Line 4), update the current vertex (Line 5), and iterate the procedure (Line 6). Both real-time search methods can be executed by ants without memory, require only a very limited look-ahead and computational capabilities, and can be used by single ants as well as groups of ants that share the u-values but do not directly communicate with each other. These properties match the limited sensing and computational capabilities of ants. In the following, we describe Learning Real-Time A* and Node Counting in detail.

*2.1. Learning Real-Time A**

Value-Update Rule of LRTA* (Line 3 in Figure 2)

$$u(s) := 1 + u(succ(s, a)).$$

Learning Real-Time A* (LRTA*) can be described as follows. The u-value $u(s)$ of vertex s approximates the sum of the smallest u-value of any state and the distance from vertex s to a closest vertex with such a smallest u-value. To understand the semantics of the u-values, consider the u-values until the graph has been covered for the first time. In this case, a vertex with u-value zero corresponds to an unvisited vertex (to be precise: a vertex that has never been left by an ant). The u-value $u(s)$ of vertex s then approximates the distance from vertex s to a closest vertex with u-value zero, that is, a closest vertex that has not yet been visited. LRTA* always moves the ants to a neighboring vertex with a smallest u-value to reach a vertex that has not yet been visited, to eventually visit all vertices and cover the graph. LRTA* sets the new u-value of the current vertex to one plus the minimum of the u-values of its neighboring vertices. The reason for this value-update rule (that can easily be generalized to graphs with

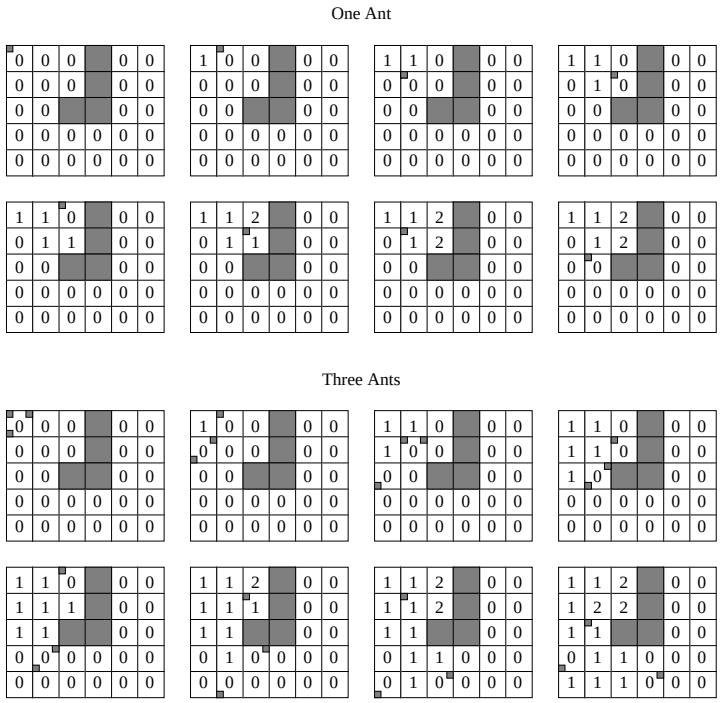


Figure 3. Example: LRTA*

non-uniform edge costs) is that the shortest path to a closest unvisited vertex has to go through one of the neighboring vertices. Thus, an approximation of the distance from the current vertex to a closest unvisited vertex is one plus the minimum of the estimates of the distances from the neighboring vertices to a closest unvisited vertex.

LRTA* was first described in [17] and is probably the most popular real-time search method. It is often used to find paths to a goal vertex on known or unknown graphs. In this case, one can initialize the u-value of a vertex with an approximation of its goal distance, which focuses the search towards the goal vertices. Furthermore, if LRTA* is always restarted when it reaches a goal vertex, then it will eventually follow a shortest path to a goal vertex. In this article, however, we apply LRTA* to graph coverage. Figure 3 demonstrates how ants that use LRTA* cover a regular four-connected grid. The ants can move to each of the four neighboring cells of their current cell provided that the destination cell is traversable (white). The ants move in a given sequential order (although this is not necessary in general). The cells are marked with their u-values. If a cell contains an ant, one of its corners is marked. Different corners represent different ants. Figure 3 (top) demonstrates how single ants that use LRTA* cover the grid, and Figure 3 (bottom) demonstrates how three ants that use LRTA* cover it.

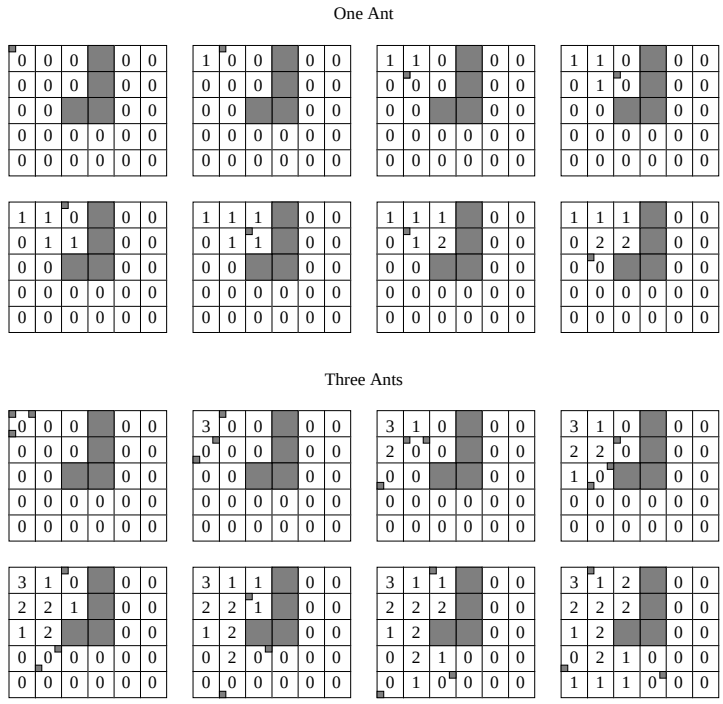


Figure 4. Example: Node Counting

2.2. Node Counting

Value-Update Rule of Node Counting (Line 3 in Figure 2)

$u(s) := 1 + u(s)$.

We compare LRTA* to Node Counting, that can be described as follows. The u-value $u(s)$ of vertex s corresponds to the number of times vertex s has been visited by ants. To understand the semantics of the u-values, consider the u-values until the graph has been covered for the first time. In this case, vertices with u-values zero correspond to unvisited vertices (to be precise: vertices that have never been left by an ant). Node Counting always moves the ants to a neighboring vertex with a smallest u-value to reach a vertex that has not yet been visited, to eventually visit all vertices and cover the graph. Node Counting sets the new u-value of the current vertex to one plus the old u-value of the current vertex (meaning that the vertex has been visited another time).

Variants of Node Counting have been used in the literature to explore unknown terrain, either on their own [20] or to accelerate reinforcement-learning methods [28]. Node Counting is also the foundation of "Avoiding the Past: A Simple but Effective Strategy for Reactive [Robot] Navigation" [2]. It is probably easier to implement ants that use Node Counting than ants that use LRTA* because Node Counting always increases the u-value of the current vertex by the

same constant, whereas LRTA* increases the u-value of the current vertex by a value that depends on the u-values of the neighboring vertices. Figure 4 demonstrates how ants that use Node Counting cover a regular four-connected grid. Figure 4 (top) demonstrates how single ants that use Node Counting cover the grid, and Figure 4 (bottom) demonstrates how three ants that use Node Counting cover it.

3. Ant Coverage with Real-Time Search

In this section, we show that ants that use LRTA* and ants that use Node Counting cover (strongly connected) graphs repeatedly and thus avoid cycling forever in parts of the graphs, despite their limited sensing and computational capabilities. We first show that ants that use LRTA* cover graphs repeatedly, following [20] and [23]. This can be shown by assuming that, at some point in time, they do not cover the graphs (again). Then, there is some (possibly later) point in time when they only visit those vertices again that they visit infinitely often; they cycle on part of the graphs. The u-values of all vertices in the cycle then increase beyond every bound since LRTA* increases the smallest u-value of the vertices in the cycle by at least one every time an ants leaves the vertex. But then the u-values of all vertices in the cycle increase above the u-values of all vertices that border the cycle. Such vertices exist according to our assumptions that the graph is strongly connected but will not be covered again. Then, however, at least one ant is forced to leave the cycle, which is a contradiction. The same argument also applies unchanged to ants that use Node Counting. This argument holds no matter in which order the ants move even if some ants move less often than others. Thus, ants that use LRTA* and ants that use Node Counting cover graphs repeatedly and can be used for one-time or continuous vacuum cleaning or lawn mowing. Furthermore, they have advantages over other search methods that also cover graphs repeatedly. For example, they are far more systematic than random walks and, different from chronological backtracking (depth-first search), can be suspended and restarted elsewhere, without even knowing where they get restarted. This is important for graph coverage with ants because sometimes the ants might get pushed accidentally to a different vertex. Most of the time they will not even realize this. Ants that use either LRTA* or Node Counting handle these situations automatically.

4. Cover Time of Real-Time Search Methods

The cover time is the time it takes for ants to cover a graph for the first time, that is, to visit all of its vertices at least once (it does not matter by which ant or ants a vertex was visited). This is important for one-time vacuum cleaning or lawn mowing since each cell has to be vacuumed or mowed at least once. In

the following sections, we study the cover time of ants that use LRTA* and ants that use Node Counting, both theoretically and experimentally.

4.1. Cover Time: Theoretical Results

In this section, we analyze the cover time (as a function of the number of vertices) that can be guaranteed by ants that use LRTA* or Node Counting. The ants share the u-values but do not directly communicate with each other. We assume that each move of an ant takes one time step and that each ant moves once during each time step. For simplicity, we assume that several ants can be at the same vertex at the same time but they can move completely asynchronously. Since LRTA* and Node Counting are algorithmically similar, it can be speculated that ants that use Node Counting guarantee a similar cover time as ants that use LRTA*. We show that the cover time of ants that use LRTA* is guaranteed to be polynomial in the number of vertices. Then, we show, surprisingly, that the cover time of ants that use Node Counting can be exponential in (the square root of) the number of vertices on both directed and undirected graphs including (planar) undirected trees. Finally, we attempt to explain the difference in cover time between ants that use LRTA* and ants that use Node Counting by presenting a condition that is sufficient for the cover time to be polynomial. Almost all of the proofs are by induction. The lower bounds require one to come up with graph topologies on which ants that use real-time search methods perform badly. This can be tricky since not much is known about real-time search methods. Instead of studying the cover time of ants directly, we study the time it takes one ant of a group of ants to reach a goal vertex for the first time, that is, any vertex in a given set $\emptyset \neq G \subseteq S$ of goal vertices. $gd(s)$ denotes the goal distance of vertex s, that is, the distance from s to a closest goal vertex. Since ants that use LRTA* and ants that use Node Counting cover graphs, they also reach a goal vertex. Their cover time in the worst case is the same as the time it takes in the worst case to reach a goal vertex for the first time (where an adversary can pick the goal vertices), for example, the vertex that was visited last when covering the graph. All of our example graphs are planar graphs, a property that is realistic for graphs that model terrains.

*4.1.1. Theoretical Results about LRTA**

In this section, we prove that the cover time of ants that use LRTA* is guaranteed to be at most $n^2 - n$, that is, polynomial in the number of vertices, no matter whether the graphs are directed or whether they are planar. But first, we study how long it takes until one ant of a group of ants that use LRTA* reaches a goal vertex for the first time. We need the following property of u-values to state our results.

Definition 1. U-values of LRTA* are admissible iff $0 \leq u(s) \leq gd(s)$ for all $s \in S$.

Admissibility means that the u-values of LRTA* never overestimate the goal distances. For example, zero-initialized u-values are admissible. Admissibility is an important concept in the context of traditional search methods such as A* [19]. We can now prove the following theorem.

Theorem 2. The u-values remain admissible after every edge traversal of ants that use LRTA* (until a goal vertex is reached for the first time) and are monotonically non-decreasing over time.

Proof. by induction, see [9]. □

We can now prove the following theorems, following [12]. Similar theorems were also independently proved in [30].

Theorem 3. Consider one particular ant of a group of ants that use LRTA*. Let s_t denote the current vertex of the ant, and s_{t+1} the vertex of the ant after it has traversed the chosen edge. Let $u_t(s)$ denote the u-value of vertex s immediately before the ant updates the u-value of s_t, and $u_{t+1}(s)$ the same u-value immediately after the update. Whenever the ant updates the u-value of its current vertex and instantaneously traverses the chosen edge, then $(2\sum_{s\in S} u_{t+1}(s) - u_{t+1}(s_{t+1})) - (2\sum_{s\in S} u_t(s) - u_t(s_t)) \geq 1$, that is, the potential $2\sum_{s\in S} u(s) - u(s_t)$ increases by at least one.

Proof. The only u-value that changes is $u(s_t)$. It increases according to Theorem 2 by $u_{t+1}(s_t) - u_t(s_t) \geq 0$. According to the value-update rule of LRTA*, it changes to $u_{t+1}(s_t) = 1 + u_t(s_{t+1})$. We distinguish two cases.

1. If $s_{t+1} \neq s_t$, then the potential increases by

$$\begin{aligned}
&(2\sum_{s\in S} u_{t+1}(s) - u_{t+1}(s_{t+1})) - (2\sum_{s\in S} u_t(s) - u_t(s_t)) \\
&= (2u_{t+1}(s_t) - u_{t+1}(s_{t+1})) - (2u_t(s_t) - u_t(s_t)) \\
&= (2u_{t+1}(s_t) - u_t(s_{t+1})) - u_t(s_t) \\
&= (2u_{t+1}(s_t) - (u_{t+1}(s_t) - 1)) - u_t(s_t) \\
&= 1 + (u_{t+1}(s_t) - u_t(s_t)) \\
&\geq 1.
\end{aligned}$$

2. If $s_{t+1} = s_t$, then the potential increases by

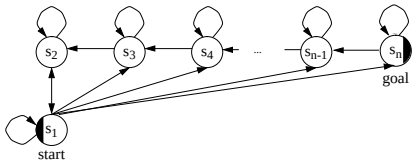


Figure 5. Single Ants that use LRTA* on Directed Graphs

$$
\begin{aligned}
&(2\sum_{s\in S} u_{t+1}(s) - u_{t+1}(s_{t+1})) - (2\sum_{s\in S} u_t(s) - u_t(s_t)) \\
&= (2u_{t+1}(s_t) - u_{t+1}(s_{t+1})) - (2u_t(s_t) - u_t(s_t)) \\
&= (2u_{t+1}(s_t) - u_{t+1}(s_t)) - (2u_t(s_t) - u_t(s_t)) \\
&= u_{t+1}(s_t) - u_t(s_t) \\
&= (1 + u_t(s_{t+1})) - u_t(s_t) \\
&= (1 + u_t(s_t)) - u_t(s_t) \\
&= 1.
\end{aligned}
$$

□

Theorem 4. The time it takes one ant of a group of ants that use LRTA* to reach a goal vertex for the first time is at most $2\sum_{s\in S} gd(s)$.

Proof. Consider one particular ant and the potential $2\sum_{s\in S} u(s) - u(s_t)$ for this ant, where s_t is the current vertex of the ant. Initially, the potential is zero. According to Theorem 3, if the ant updates the u-value of its current vertex and traverses the chosen edge, the potential increases by at least one. According to Theorem 2, if a different ant updates the u-value of its current vertex, the potential cannot decrease. Finally, according to Theorem 2, the potential is bounded from above by $2\sum_{s\in S} gd(s)$. Consequently, at least one ant of a group of ants that use LRTA* reaches a goal vertex for the first time after at most $2\sum_{s\in S} gd(s)$ edge traversals. □

Theorem 4 states that the time it takes one ant of a group of ants that use LRTA* to reach a goal vertex for the first time is at most $2\sum_{s\in S} gd(s)$. Thus, the fewer vertices there are and the smaller the average goal distance over all vertices, the smaller the upper bound. This corresponds to our intuition. For example, if all vertices are clustered around the goal vertices, then each ant can never move far away from the goal vertices and always has a chance of getting to one quickly even if it behaved suboptimally in the past. It holds that $2\sum_{s\in S} gd(s) \leq 2\sum_{i=0}^{n-1} i = n^2 - n$ according to our assumption that the graphs are strongly connected. In the following, we present an example graph that shows that this upper bound is tight for single ants that use LRTA*. Consider the graph in Figure 5. The following program in pseudo code shows one possible vertex sequence that single ants that use LRTA* can traverse. The scope of

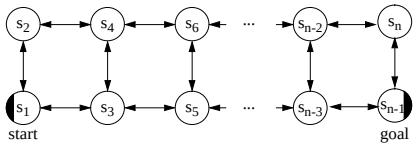


Figure 6. Single Ants that use LRTA* on Undirected Graphs

the for-statements is shown by indentation. Ties are broken by remaining in the current vertex if possible and otherwise choosing the vertex with the lowest label.

```
for i := 1 to n-1
  for j := 1 to i
    print s(i)
  for j := i downto 1
    print s(j)
print s(n)
```

This can be proved by induction on n. It is then easy to determine from an inspection of the pseudo code that single ants traverse $n^2 - n$ edges before they reach the goal vertex for the first time, proving that the upper bound of $n^2 - n$ edge traversals is tight. For example, for $n = 5$, they traverse the vertex sequence s_1, s_1, s_2, s_2, s_2, s_1, s_3, s_3, s_3, s_3, s_2, s_1, s_4, s_4, s_4, s_4, s_3, s_2, s_1, and s_5, and thus 20 edges. The graph used in the above example was artificially constructed. However, the upper bound of $O(n^2)$ edge traversals is tight even for more realistic graphs such as regular four-connected grids. Consider the grid shown in Figure 6 and assume that $n \geq 2$ with $n \bmod 4 = 2$. The following program in pseudo code shows one possible vertex sequence that single ants that use LRTA* can traverse.

```
for i := n-3 downto n/2 step 2
  for j := 1 to i step 2
    print s(j)
  for j := i+1 downto 2 step 2
    print s(j)
for i := 1 to n-1 step 2
  print s(i)
```

This can be proved by induction on n. It is then easy to determine that single ants traverse $3n^2/16 - 3/4$ edges before they reach the goal vertex for the first time, proving that the upper bound of $O(n^2)$ edge traversals is tight even for undirected planar graphs without edges that leave the current vertex unchanged and for which the number of outgoing edges of any vertex is bounded from above by a small constant (here: three). For example, for $n = 10$, they traverse the vertex sequence s_1, s_3, s_5, s_7, s_8, s_6, s_4, s_2, s_1, s_3, s_5, s_6, s_4, s_2, s_1, s_3, s_5, s_7, and s_9, and thus 18 edges.

Since the time it takes one ant of a group of ants in the worst case to reach a goal vertex for the first time is an upper bound on the cover time in the worst case, the following corollary follows from Theorem 4.

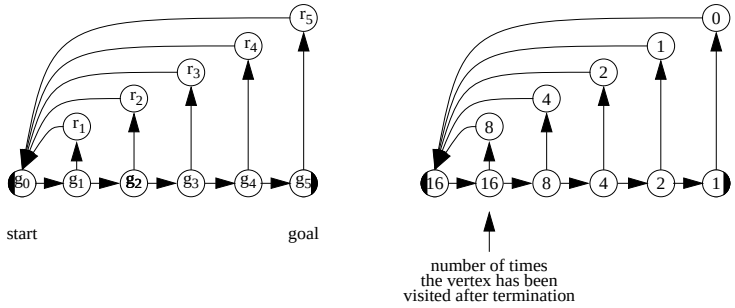


Figure 7. Single Ants that use Node Counting on Directed Graphs ($m = 5, n = 11$)

Corollary 5. The cover time of ants that use LRTA* on strongly connected (directed or undirected, planar or non-planar) graphs is guaranteed to be polynomial in the number of vertices.

4.1.2. Theoretical Results about Node Counting

In this section, we prove that the cover time of ants that use Node Counting can be exponential in (the square root of) the number of vertices for both directed and undirected graphs, including (planar) undirected trees. First, we discuss directed graphs. Then, we discuss undirected graphs, a special case of directed graphs since every undirected edge is modeled with two directed edges of opposite directions. We first discuss directed graphs because our directed example graphs are simpler than our undirected example graphs. In both cases, we first study how long it takes until a single ant that uses Node Counting reaches a goal vertex for the first time. Then, we generalize to how long it takes until one ant of a group of ants that use Node Counting reaches a goal vertex for the first time. Finally, we generalize to the cover time of ants that use Node Counting.

Cover Time of Node Counting on Directed Graphs: In this section, we present a directed example graph that shows that the number of edge traversals until a single ant that uses Node Counting reaches a goal vertex for the first time can be exponential in the number of vertices. Figure 7 shows an instance of this example graph. In general, the graphs have $m + 1$ levels (for $m \geq 1$). The levels consist of vertices of two different kinds: g-vertices and r-vertices. At level $i = 0$, there is only one vertex, namely a g-vertex g_0. This vertex is connected to g-vertex g_1. At levels $i = 1 \ldots m$, there are two vertices, namely a g-vertex g_i and an r-vertex r_i. The g-vertex g_i is connected to g-vertex g_{i+1} (if $i \neq m$) and r-vertex r_i. The r-vertex r_i is connected to g-vertex g_0. The start vertex is g_0 and the goal vertex is g_m. The graphs have $n = 2m + 1$ vertices. Single ants that use Node Counting can traverse the vertex sequence that is printed by $f(m)$ of the following program in pseudo code. Ties are broken towards r-vertices.

```
proc f(i) =
  if i = 1 then
```

m	number of vertices	Node Counting		LRTA*	
		edge traversals	$\frac{\text{edge traversals}}{\text{number of vertices}}$	edge traversals	$\frac{\text{edge traversals}}{\text{number of vertices}}$
1	3	1	0.3	1	0.3
2	5	5	1.0	5	1.0
3	7	13	1.9	10	1.4
4	9	29	3.2	16	1.8
5	11	61	5.5	26	2.4
6	13	125	9.6	34	2.6
7	15	253	16.9	43	2.9
8	17	509	29.9	60	3.5
...	...	...	...	...	...
20	41	2097149	51150.0	321	7.8
21	43	4194301	97541.9	344	8.0
22	45	8388605	186413.4	378	8.4
23	47	16777213	356962.0	433	9.2

Table 1
Results for Node Counting and LRTA*

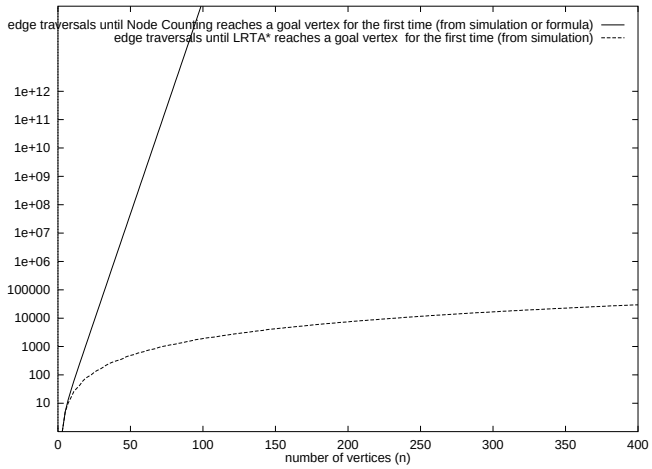


Figure 8. Results (Log Scale!)

```
  print g(0)
else
  f(i-1)
  print r(i-1)
  f(i-1)
print g(i)
```

This can be proved by induction on m. It is then easy to determine that single ants traverse $2^{n/2+1/2}-3$ edges before they reach the goal vertex for the first time, proving that the number of edge traversals until the goal vertex is reached for the first time can be exponential in the number of vertices. Theorem 6 follows. For example, for $m = 5$ and $n = 11$, they traverse the vertex sequence g_0, g_1, r_1, g_0, g_1, g_2, r_2, g_0, g_1, r_1, g_0, g_1, g_2, g_3, r_3, g_0, g_1, r_1, g_0, g_1, g_2, r_2, g_0, g_1, r_1, g_0, g_1, g_2, g_3, g_4, r_4, g_0, g_1, r_1, g_0, g_1, g_2, r_2, g_0, g_1, r_1, g_0, g_1, g_2, g_3, r_3, g_0, g_1, r_1, g_0, g_1, g_2, r_2, g_0, g_1, r_1, g_0, g_1, g_2, g_3, g_4, and g_5, and thus 61 edges. This is consistent with Figure 7 since summing up the number of times that the vertices in the figure have been visited until a goal vertex is reached for the first time (except for the goal vertex) yields 61. The reason for the large number of

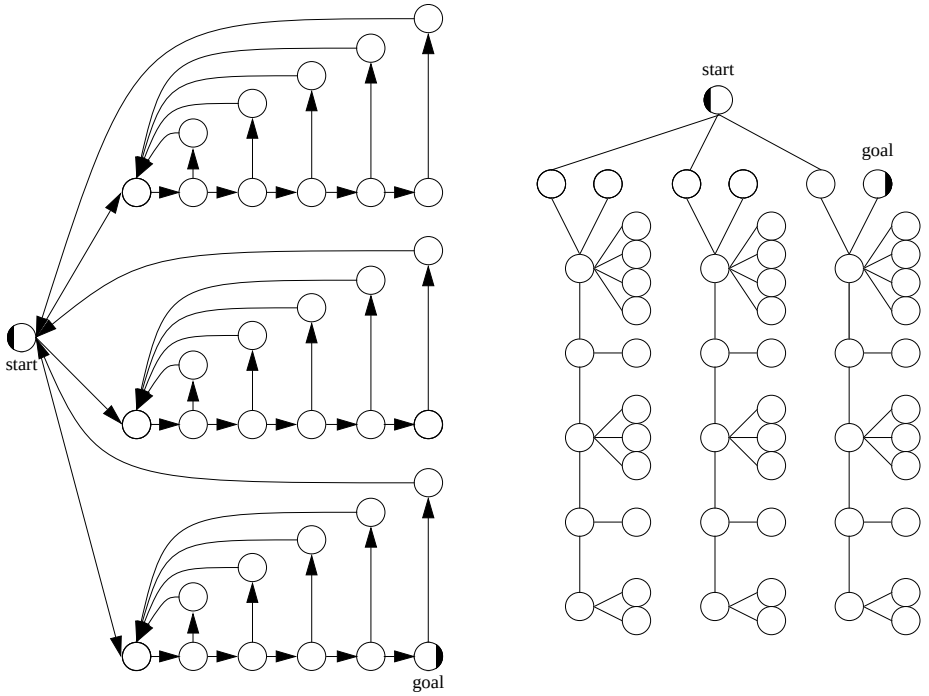


Figure 9. Ants that use Node Counting

edge traversals is the large number of ties between the successor g-vertex and the successor r-vertex when the ant is at a g-vertex. Since ties are broken towards r-vertices, the ant moves away from the goal vertex. The u-value of the r-vertex is subsequently increased by only one, which soon results in another tie between the g-vertex and the r-vertex. This could be avoided by increasing the u-value of the r-vertex by more than one, which is what happens if the ant uses LRTA* instead of Node Counting. Table 1 and Figure 8 show results for single ants that use Node Counting and single ants that use LRTA*. In both cases, the ants break ties towards r-vertices.

Theorem 6. The time it takes a single ant that uses Node Counting to reach a goal vertex for the first time on strongly connected (directed) graphs can be exponential in the number of vertices even if the graphs are planar.

Proof. by induction on the number of vertices. □

This theorem can easily be extended to groups of k ants (for any constant k). In this case, we replicate our graph k times, see Figure 9 (left). We introduce a new start vertex, with (directed) edges going from it to all old start vertices. The new goal vertex is (any) one of the old goal vertices. Finally, we redirect all edges from the successor vertices of the old goal vertices to the new start vertex.

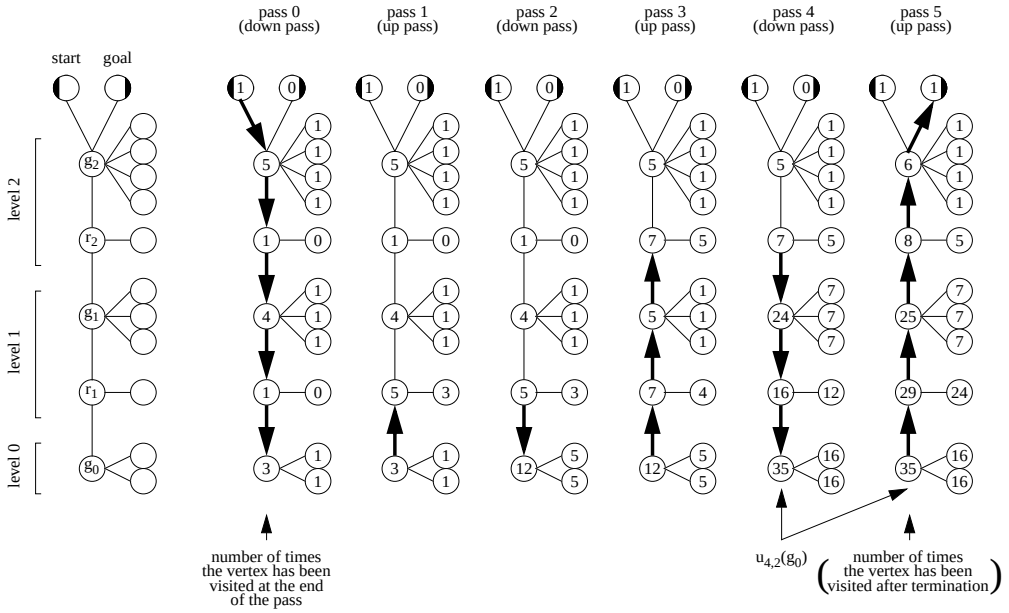


Figure 10. Single Ants that use Node Counting on Undirected Graphs ($m = 2, n = 18$)

If k ants that use Node Counting are started in the new start vertex, each ant can move to a different one of the old start vertices and then exhibit the behavior described above in the context of single ants. Since the number of vertices of the new example graph is only a constant factor larger than the number of vertices of the old example graph, the following theorem follows.

Corollary 7. The time it takes a given number of ants that use Node Counting to reach a goal vertex for the first time on strongly connected (directed) graphs can be exponential in the number of vertices even if the graphs are planar.

Since the time it takes a group of ants in the worst case to reach a goal vertex for the first time is a lower bound on the cover time in the worst case, the following corollary follows.

Corollary 8. The cover time of a given number of ants that use Node Counting on strongly connected (directed) graphs can be exponential in the number of vertices even if the graphs are planar.

Cover Time of Node Counting on Undirected Graphs: Graphs that correspond to terrain are often undirected, including grids. It is conceivable that ants that use Node Counting are more efficient on undirected graphs than directed graphs and thus that their cover time might always be polynomial in the number of vertices on

undirected graphs. In this section, however, we present an undirected example graph that shows that their cover time can still be exponential in (the square root of) the number of the vertices even for (planar) undirected trees. Despite its elegance, this example graph proved to be rather difficult to construct. We first study the number of edge traversals until single ants that use Node Counting reach a goal vertex for the first time. Figure 10 shows an instance of our example graph. In general, the trees have $m + 1$ levels (for $m \geq 2$). The levels consist of vertices of three different kinds: g-subroots, r-subroots, and leafs that are connected to the subroots. g-subroots and r-subroots alternate. At level $i = 0$, there is one subroot, namely a g-subroot g_0. At levels $i = 1 \ldots m$, there are two subroots, namely an r-subroot r_i and a g-subroot g_i. Subroot g_i has $m + i$ leafs connected to it, and subroot r_i has one leaf connected to it. Finally, subroot g_m is connected to two additional vertices, namely the start vertex and the only goal vertex. The trees have $n = 3/2\ m^2 + 9/2\ m + 3$ vertices. Single ants that use Node Counting proceed in a series of passes through the trees. During each pass, they traverse the subroots in the opposite order than the previous pass. We call a pass during which they traverse the subroots in descending order a down pass, and a pass during which they traverse them in ascending order an up pass. We number passes from zero on upward, so even passes are down passes and odd passes are up passes. A pass ends immediately before it switches directions. We break ties as follows: During pass zero, ties among successor vertices are broken in favor of leaf vertices of g-subroots (with highest priority) and then subroots. Pass zero ends when the single ants have visited the leafs of subroot g_0 once each. As a result, at the end of pass zero they have visited the subroots g_i $m + i + 1$ times each and their leafs once each. They have visited the subroots r_i once each and their subroots not at all. During all subsequent passes, ties among successor vertices are broken in favor of subroots whenever possible. A tie between two r-subroots (when the single ants are at a g-subroot) is resolved by continuing with the current pass. A tie between two g-subroots (when the single ants are at an r-subroot) is resolved by terminating the current pass and starting a new one in the opposite direction. In the following, we provide a sketch of the proof that single ants that use Node Counting traverse (at least)

$$\Omega(n^{\sqrt{(\frac{1}{6}-\epsilon)n}})$$

edges in this case, where n is the number of vertices and $0 < \epsilon < 1/6$ is an arbitrarily small constant.

We use the following notation for the proof. Let $u_{p,m}(s)$ denote the total number of times that subroot s has been entered at the end of pass p for a tree with $m+1$ levels. By definition, $u_{p,m}(s)$ is a nondecreasing function of p. Our tie breaking rules guarantee that all leafs of a subroot have been entered the same number of times at the end of each pass, so we let $w_{p,m}(s)$ denote the number of times that each of the leafs of subroot s has been entered at the end of pass

p for a tree with $m+1$ levels. Finally, let $x_{p,m}(s)$ denote the total number of times that subroot s has been entered from non-leafs at the end of pass p for a tree with $m+1$ levels. These values relate as follows: The total number of times that a subroot was entered at the end of pass p is equal to the product of the number of its leafs and the total number of times that it was entered from each of its leafs at the end of pass p (which equals the total number of times that each of its leafs was entered at the end of pass p) plus the total number of times that the subroot was entered from non-leafs at the end of pass p. For example, $u_{p,m}(g_i) = (m+i)\,w_{p,m}(g_i) + x_{p,m}(g_i)$.

We now outline the idea of the proof. Pass 0 is the first pass during which the single ants move from subroot r_1 to subroot g_0. The u-value of subroot g_0 at the end of pass 0 is $u_{0,m}(g_0) = m+1$. In the following, we show that every subsequent pass during which the single ants move from subroot r_1 to subroot g_0 at least doubles the u-value of subroot g_0. Thus, the total number of edge traversals grows exponentially in the number of down passes that move the single ants from subroot r_1 to subroot g_0. For the proof, assume that the single ants move from subroot r_1 to subroot g_0 during down pass $p > 0$. At this point in time, the u-value of subroot g_0 must be less than or equal to the u-value of the leaf of subroot r_1, otherwise the single ants could not move to subroot g_0 ($u_{p-1,m}(g_0) \leq w_{p,m}(r_1)$). Furthermore, at this point in time, the u-value of the leaf of subroot r_1 must be less than or equal to the u-value of subroot r_1, because the single ants have to move to subroot r_1 whenever they are in its leaf ($w_{p,m}(r_1) \leq u_{p,m}(r_1)$). The down pass is over when the single ants switch direction and move from subroot g_0 back to subroot r_1. At this point in time, the u-value of subroot r_1 must be less than or equal to the u-value of each of the leafs of subroot g_0, otherwise the single ants could not move to subroot r_1 ($u_{p,m}(r_1) \leq w_{p,m}(g_0)$). Furthermore, at this point in time, m times the u-value of each of the leafs of subroot g_0 must be less than or equal to the u-value of subroot g_0, because the single ants have to move to subroot g_0 whenever they are in one of its m leafs ($m\,w_{p,m}(g_0) \leq u_{p,m}(g_0)$). Put together: $m u_{p-1,m}(g_0) \leq m\,w_{p,m}(r_1) \leq m\,u_{p,m}(r_1) \leq m\,w_{p,m}(g_0) \leq u_{p,m}(g_0)$ for $m \geq 2$, which proves that every pass during which the single ants move from subroot r_1 to subroot g_0 at least doubles the u-value of subroot g_0. Thus, the total number of edge traversals grows exponentially in the number of down passes that move the single ants from subroot r_1 to subroot g_0. To ensure that this number is large, we need to ensure that the number of down passes is large and thus that many up passes end before the goal vertex is reached. In the following, we show that every time an up pass reaches a subroot r_{i+1} for the second time, the up pass ends. Pass 0 is the first pass during which the single ants visit subroot g_i for $1 \leq i < m$. Assume that the second pass during which the single ants visit subroot g_i is pass p. This pass must be an up pass since pass 0 is a down pass and ends at subroot g_0. In the following, we show that the single ants, after they have moved from subroot r_i to subroot g_i for the first time, continue to move to subroot r_{i+1} and then end the up pass and start a down pass. Thus, every

time an up pass reaches a subroot r_{i+1} for the second time, the up pass ends. Consequently, there are a large number of up and thus also down passes. For the proof, notice that, at the end of pass 0, the u-values of subroot r_i, subroot r_{i+1}, and the leafs of subroot g_i are one. The u-value of subroot g_i is $m+i+1$, and the u-value of subroot g_{i+1} is $m+i+2$. Now assume that the single ants move from subroot r_i to subroot g_i for the first time. At this point in time, subroot r_{i+1} and the leafs of subroot g_i have not been visited again since pass 0 and their u-values are thus still one. Subroot g_i has been visited only one time after pass 0 and its u-value is thus $m+i+2$. Subroot r_i has been visited at least once after pass 0 and thus its u-value is at least two. According to our tie breaking rules, the up pass p continues and the single ants move from subroot g_i to subroot r_{i+1}. At this point in time, the u-value of subroot g_i is still $m+i+2$ (see above). Subroot g_{i+1} has not been visited again since pass 0 and its u-value is thus still $m+i+2$. According to our tie breaking rules, the next subroot that the ants move to is subroot g_i. Thus, the up pass ends and the single ants move towards subroot g_0 again. Consequently, there are a large number of up and thus also down passes. The purpose of the particular topology of the graph then is to ensure that the single ants move from subroot r_1 to subroot g_0 during each down pass and do not stop the down pass earlier. In the following, we provide a sketch of the proof that this is indeed the case.

Lemma 9. Assume that single ants that use Node Counting visit subroot s (with $s \neq g_m$) during pass p, where $0 < p < 2m+2$. The values $u_{p,m}(s)$ can then be calculated as follows, for $i > 0$:

$$
\begin{aligned}
u_{2k+1,m}(g_0) &= m\ u_{2k,m}(r_1) + x_{2k,m}(g_0) \\
u_{2k,m}(g_0) &= m\ u_{2k,m}(r_1) + x_{2k,m}(g_0)
\end{aligned}
$$

$$
\begin{aligned}
u_{2k,m}(g_i) &= (m+i)\min(u_{2k-1,m}(r_i), u_{2k,m}(r_{i+1})) + x_{2k,m}(g_i) \\
u_{2k+1,m}(g_i) &= (m+i)\min(u_{2k,m}(r_{i+1}), u_{2k+1,m}(r_i)) + x_{2k+1,m}(g_i) \\
u_{2k,m}(r_i) &= \min(u_{2k-1,m}(g_{i-1}), u_{2k,m}(g_i)) + x_{2k,m}(r_i) \\
u_{2k+1,m}(r_i) &= \min(u_{2k,m}(g_i), u_{2k+1,m}(g_{i-1})) + x_{2k+1,m}(r_i)
\end{aligned}
$$

Proof. by induction on the number of passes p for the example graphs, see [26]. □

Theorem 10. If $p = 2k$ for $0 \leq k \leq m$, then the down pass ends at subroot g_0 and it holds that

$$u_{2k,m}(g_i) = \begin{cases} m(u_{2k-2,m}(g_i)+2k)+k+1 & \text{for } i=0<k \\ (m+i)(u_{2k-2,m}(g_i)+2k-2i \\ \quad +1)+2k-2i+1 & \text{for } 0<i<k \\ m+i+1 & \text{otherwise} \end{cases}$$

$$u_{2k,m}(r_i) = \begin{cases} u_{2k-1,m}(g_{i-1})+2k-2i+2 & \text{for } 0<i\le k \\ 1 & \text{otherwise} \end{cases}$$

$$x_{2k,m}(g_i) = \begin{cases} k+1 & \text{for } i=0 \\ 2k-2i+1 & \text{for } 0<i<k \\ 1 & \text{otherwise} \end{cases}$$

$$x_{2k,m}(r_i) = \begin{cases} 2k-2i+2 & \text{for } 0<i\le k \\ 1 & \text{otherwise} \end{cases}$$

$$\begin{aligned} u_{2k,m}(r_i) &\ge u_{2k-1,m}(r_{i-1}) && \text{for } 1<i\le k \\ u_{2k,m}(g_i) &> u_{2k-1,m}(g_{i-1}) && \text{for } 0<i<k \end{aligned}$$

If $p=2k+1$ for $0\le k\le m$, then the up pass ends at subroot r_{k+1} (with the exception of up pass $2m+1$ that ends at the goal vertex) and it holds that

$$u_{2k+1,m}(g_i) = \begin{cases} u_{2k,m}(g_i) & \text{for } i=0 \\ (m+i)(u_{2k-1,m}(g_i)+2k \\ \quad -2i)+2k-2i+2 & \text{for } 0<i<k \\ m+i+2 & \text{for } 0<i=k \\ m+i+1 & \text{otherwise} \end{cases}$$

$$u_{2k+1,m}(r_i) = \begin{cases} u_{2k,m}(g_i)+2k-2i+3 & \text{for } 0<i\le k \\ u_{2k+1,m}(g_{i-1})+2 & \text{for } i=k+1\le m \\ 1 & \text{otherwise} \end{cases}$$

$$x_{2k+1,m}(g_i) = \begin{cases} k+1 & \text{for } i=0 \\ 2k-2i+2 & \text{for } 0<i\le k \\ 1 & \text{otherwise} \end{cases}$$

$$x_{2k+1,m}(r_i) = \begin{cases} 2k-2i+3 & \text{for } 0<i\le k \\ 2 & \text{for } i=k+1\le m \\ 1 & \text{otherwise} \end{cases}$$

$$\begin{aligned} u_{2k+1,m}(r_i) &\ge u_{2k,m}(r_{i+1}) && \text{for } 0<i\le k \\ u_{2k+1,m}(g_i) &> u_{2k,m}(g_{i+1}) && \text{for } 0\le i<k \end{aligned}$$

Proof. by induction on the number of traversed edges for the example graphs, using Lemma 9, see [26]. □

Thus, there are $2m+2$ passes until a goal vertex is reached for the first time. Each down pass ends at subroot g_0. The final up pass ends at the goal vertex. All other up passes p end at subroot $r_{(p+1)/2}$. In the following, we need two inequalities. First, according Theorem 10, it holds for $0\le k\le m$ that

$$u_{2k,m}(g_0) = \begin{cases} m+1 & \text{for } k = 0 \\ m(u_{2k-2,m}(g_0) + 2k) + k + 1 & \text{otherwise.} \end{cases}$$

Solving the recursion yields

$$u_{2k,m}(g_0) = \frac{m^{k+3} + m^{k+2} + m^{k+1} - (2k+2)m^2 + (k-2)m + k + 1}{m^2 - 2m + 1}.$$

Setting $k = m$ in this formula results in

$$u_{2m,m}(g_0) = \frac{m^{m+3} + m^{m+2} + m^{m+1} - 2m^3 - m^2 - m + 1}{m^2 - 2m + 1}.$$

This implies that $u_{2m+1,m}(g_0) = u_{2m,m}(g_0) > m^m$. For example, $u_{4,2}(g_0) = 35$, which is consistent with Figure 10.

Second, consider an arbitrary constant $0 < \epsilon < 1/6$ and assume that $m > \max\left(\frac{1}{\epsilon} - 4, \left(\frac{3}{2-8\epsilon}\right)^{1/\epsilon}\right) \geq 2$. Notice that $n \geq m$ for our trees. Then,

$$\begin{aligned} n &= \tfrac{3}{2}m^2 + \tfrac{9}{2}m + 3 \\ &< \tfrac{3}{2}\left(1 + \tfrac{4}{m}\right)m^2 \quad \text{since } m > 2 \\ &< \tfrac{3}{2}\left(1 + \tfrac{4\epsilon}{1-4\epsilon}\right)m^2 \text{ since } m > \tfrac{1}{\epsilon} - 4 \\ &= \tfrac{3}{2}\tfrac{m^2}{1-4\epsilon} \end{aligned}$$

and thus $m > \sqrt{\frac{2}{3}n\,(1-4\epsilon)} > 1$ (A). In the following, we also utilize that $(an)^k > n^{(1-\epsilon)k}$ for $n > \left(\frac{1}{a}\right)^{1/\epsilon}$ and arbitrary constants $a > 0$ and $k > 0$ (B). Then,

$$\begin{aligned} m^m &> \left(\sqrt{\tfrac{2}{3}n\,(1-4\epsilon)}\right)^{\sqrt{\frac{2}{3}n(1-4\epsilon)}} \text{ since (A)} \\ &= \left(\left(\tfrac{2}{3} - \tfrac{8}{3}\epsilon\right)n\right)^{\frac{1}{2}\sqrt{\frac{2}{3}n(1-4\epsilon)}} \\ &> n^{(1-\epsilon)\frac{1}{2}\sqrt{\frac{2}{3}n(1-4\epsilon)}} \qquad \text{since (B)} \\ &= n^{\sqrt{(1-\epsilon)^2\frac{1}{6}n(1-4\epsilon)}} \\ &> n^{\sqrt{(1-2\epsilon)\frac{1}{6}n(1-4\epsilon)}} \\ &= n^{\sqrt{\left(\frac{1}{6}-\epsilon+\frac{4}{3}\epsilon^2\right)n}} \\ &> n^{\sqrt{\left(\frac{1}{6}-\epsilon\right)n}} \end{aligned}$$

Using the two inequalities above, it holds that $u_{2m+1,m}(g_0) > m^m > n^{\sqrt{\left(\frac{1}{6}-\epsilon\right)n}}$ for $m > \max\left(\frac{1}{\epsilon} - 4, \left(\frac{3}{2-8\epsilon}\right)^{1/\epsilon}\right)$ and thus also for sufficiently large

m	number of vertices n	number of passes	u-value of g_0 $u_{2m+1,m}(g_0)$	$\frac{\text{u-value of } g_0}{\text{number of vertices}}$	edge traversals	$\frac{\text{edge traversals}}{\text{number of vertices}}$
2	18	6	35	1.9	190	10.6
3	30	8	247	8.2	1380	46.0
4	45	10	2373	52.7	12330	274.0
5	63	12	30256	480.3	142318	2259.0
6	84	14	481471	5731.8	2063734	24568.3
7	108	16	9127581	84514.6	36135760	334590.4
8	135	18	199957001	1481163.0	740474450	5484995.9

Table 2
Results for Node Counting

m	number of vertices n	number of passes	edge traversals	$\frac{\text{edge traversals}}{\text{number of vertices}}$
2	18	2	32	1.8
3	30	2	56	1.9
4	45	2	86	1.9
5	63	2	122	1.9
6	84	2	164	2.0
7	108	2	212	2.0
8	135	2	266	2.0

Table 3
Results for LRTA*

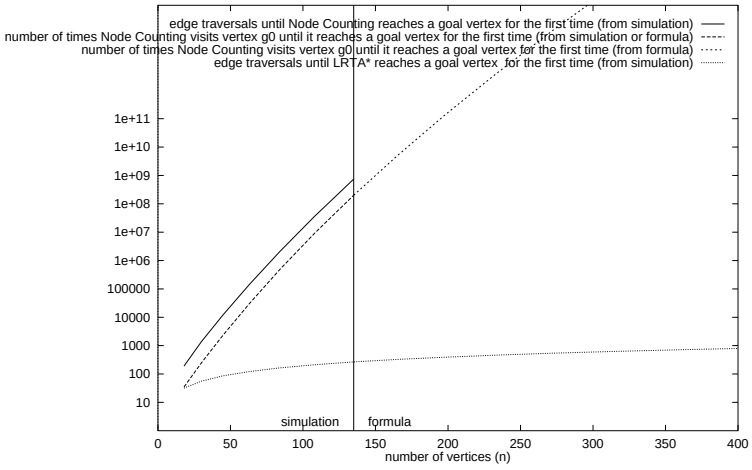


Figure 11. Results (Log Scale!)

n. $u_{2m+1,m}(g_0)$ is the u-value of g_0 after single ants that use Node Counting reach a goal vertex for the first time on a tree with $m+1$ levels. This u-value equals the number of times that the single ants have visited g_0, which is a lower bound on the number of edges they have traversed until a goal vertex is reached for the first time. Consequently, single ants that use Node Counting traverse (at least)

$$\Omega(n^{\sqrt{(\frac{1}{6}-\epsilon)n}})$$

edges, and the number of edge traversals until a goal vertex is reached for the first time can be exponential in (the square root of) the number of vertices even

for (planar) undirected trees. Theorem 11 follows. For comparison, single ants that use LRTA* visit each leaf of a g-subroot once while they move in one down pass from the start vertex to g-subroot g_0 (if ties are broken as before). Then, they visit each leaf of an r-subroot once while they move in one up pass from the g-subroot g_0 to the goal vertex, for a total of $3m^2 + 9m + 2 = 2n - 4$ edge traversals.

Theorem 11. The time it takes single ants that use Node Counting to reach a goal vertex for the first time on strongly connected undirected graphs can be exponential in (the square root of) the number of vertices even for (planar) undirected trees.

Proof. by induction on the number of traversed edges for the example graphs using Lemma 9 and Theorem 10. □

To confirm this theoretical result, we performed a study of single ants that use Node Counting on our example graphs, see Table 2. We report only results for small m because the number of edge traversals grows quickly. These results confirmed our formulas for how the number of vertices n, the number of passes until a goal vertex is reached for the first time, and the u-value of g_0 when a goal vertex is reached for the first time depend on m. They also provided us with the number of edge traversals until a goal vertex is reached for the first time, for which we do not have a formula in closed form, only a lower bound in form of the u-value of g_0 when a goal vertex is reached for the first time. How the number of edge traversals and its lower bound relate is shown in Figure 11. Visual inspection of the graphs suggests that the lower bound is approximately at a constant distance from the number of edge traversals in the log plot, suggesting that the lower bound underestimates the number of edge traversals by roughly a constant factor. For comparison, Table 3 and Figure 11 contain results for LRTA* on the example graphs (if ties are broken as before).

Theorem 11 can easily be extended to groups of k ants (for any constant k). In this case, we replicate our example graph k times, see Figure 9 (right). We introduce a new start vertex, with (undirected) edges between it and all old start vertices. The new goal vertex is (any) one of the old goal vertices. If k ants that use Node Counting are started in the new start vertex, each ant can move to a different one of the old start vertices and then exhibit the behavior described above in the context of single ants. Since the number of vertices of the new example graph is only a constant factor larger than the number of vertices of the old example graph, the following theorem follows.

Corollary 12. The time it takes a given number of ants that use Node Counting to reach a goal vertex for the first time on strongly connected undirected graphs

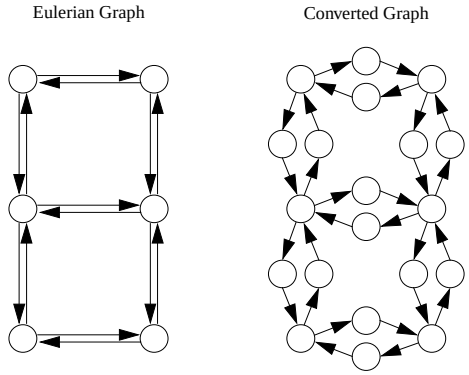


Figure 12. Sample Eulerian Graph and its Conversion

can be exponential in (the square root of) the number of vertices even for (planar) undirected trees.

Since the time it takes a group of ants in the worst case to reach a goal vertex for the first time is a lower bound on the cover time in the worst case, the following corollary follows.

Corollary 13. The cover time of a given number of ants that use Node Counting on strongly connected undirected graphs can be exponential in (the square root of) the number of vertices even for (planar) undirected trees.

Cover Time of Node Counting for Special Cases: So far, we have shown that the cover time of ants that use Node Counting is not guaranteed to be polynomial in the number of vertices for either directed or undirected graphs, including (planar) undirected trees. We are also able to describe a graph property that guarantees a polynomial cover time of single ants that use Node Counting. A Eulerian graph is a graph that contains a Eulerian tour. A Eulerian tour is a path whose start vertex equals its end vertex and that contains all of the edges of the graph exactly once [7]. This implies for directed Eulerian graphs that each vertex has an equal number of incoming and outgoing directed edges. Now consider an arbitrary strongly connected directed Eulerian graph whose number of edges is at most polynomial in the number of its vertices (for example, because no two edges connect the same vertices in the same direction), and a graph that is derived from the directed Eulerian graph by replacing each of the directed edges with two directed edges that are connected with a (unique) intermediate vertex. Figure 12 shows an example. Using results from [13], we can show that the performance of single ants that use Node Counting on the converted graph is guaranteed to be polynomial in its number of its vertices. This is so because the behavior of single ants that use Node Counting on the converted graph is the same as the behavior of single ants that use Edge Counting [13] on the original graph,

a real-time search method that always makes the ant traverse an edge that has been traversed the least number of times, and the performance of single ants that use Edge Counting on strongly connected directed Eulerian graphs is polynomial in the product of the number of its edges and the number of its vertices [13]. It is currently unknown whether grids and other graph topologies that represent realistic terrains guarantee ants that use Node Counting a polynomial cover time.

4.1.3. Discussion of the Theoretical Results

So far, we have shown that the cover time of single ants that use Node Counting can be exponential in (the square root of) the number of vertices even for (planar) undirected trees. While we realized early on that the cover time of single ants that use Node Counting is exponential on directed graphs, we tried to prove for a long time that it was polynomial on undirected graphs. That we eventually proved the opposite was surprising to us because the value-update rule of LRTA* is similar to that of Node Counting but guarantees single ants a cover time that is polynomial in the number of vertices instead of exponential. There also exist other real-time search methods whose cover time on directed or undirected graphs is polynomial in the number of vertices (perhaps with the restriction that the graphs have no edges that leave the current vertex unchanged). Table 4 shows some of these real-time search methods. Wagner, Lindenbaum and Bruckstein's Value-Update Rule[1] [29] (short: Wagner's Value-Update Rule) and Thrun's Value-Update Rule [27] resemble Node Counting even more closely than LRTA* resembles Node Counting since their value-update rules contain the term $1 + u(s)$ just like Node Counting. Their cover times were analyzed by their authors and shown to be polynomial in the number of vertices for single ants. We have shown that the cover time of Node Counting can be exponential in the number of vertices for single ants. Consequently, these small modifications of Node Counting are necessary to guarantee a polynomial cover time.

Table 4
Different Real-Time Search Methods

Value-Update Rules WITHOUT Polynomial Cover Time Guarantee	
$u(s) := 1 + u(s)$.	Node Counting
Value-Update Rules WITH Polynomial Cover Time Guarantee	
$u(s) := 1 + u(succ(s, a))$.	LRTA*
if $u(s) \leq u(succ(s, a))$ then $u(s) := 1 + u(s)$.	Wagner's Value-Update Rule [29]
$u(s) := \max(1 + u(s), 1 + u(succ(s, a)))$.	Thrun's Value-Update Rule [27]

[1] The authors call this real-time search method Vertex-Ant Walk. We do not use this name here because they call one real-time search method Vertex-Ant Walk in [29] and a different one Vertex-Ant Walk in [30]. We refer to the real-time search method described in [29]. The other real-time search method is a version of LRTA*.

We now attempt to give an intuition why the cover time of ants that use some real-time search methods can be exponential in the number of vertices while the cover time of ants that use other real-time search methods is guaranteed to be polynomial. We do this in form of a guideline for how to design real-time search methods that result in a polynomial cover time. For simplicity, we discuss only the case where single ants have to reach a goal vertex for the first time. As argued earlier, the time it takes single ants in the worst case to reach a goal vertex for the first time is the same as the cover time of single ants in the worst case. The results can then be generalized to groups of ants, which we do elsewhere.

Theorem 14. Let s_t denote the current vertex of a single ant at point in time t, and s_{t+1} the vertex of the ant after it has traversed the chosen edge. Let $u_t(s)$ denote the u-value of vertex s immediately before the ant updates the u-value of s_t, and $u_{t+1}(s)$ the same u-value immediately after the update. The time it takes the single ant to reach a goal vertex on strongly connected (directed or undirected, planar or non-planar) graphs for the first time is at most $(f(n)+1)f(n)\sum_{s\in S} gd(s)$ if all of the following conditions hold.

Condition 1: At every point in time, the u-values $u(s)$ are integers.

Condition 2: At every point in time t, it holds that $u_t(s_t) \leq u_{t+1}(s_t)$.

Condition 3: At every point in time t where $u_t(s_t) \leq \min_{a\in A(s_t)} u_t(succ(s_t,a))$, it holds that $u_t(s_t)+1 \leq u_{t+1}(s_t)$.

Condition 4: At every point in time, it holds that $|u(s)-u(succ(s,a))| \leq f(n)$ for all non-goal vertices $s \in S$ and $a \in A(s)$, where $f(n) \geq 1$ is a function of the number of vertices n.

Proof. The proof is in three parts.

1. First, we prove by induction on the number of time steps t that, at every point in time, it holds that $u(s) \leq f(n)gd(s)$ for all vertices s. It holds at time step $t=0$ since then $u(s)=0$ for all vertices s. Suppose that the induction hypothesis holds at time step t. The only u-value that changes at time step t is $u(s_t)$ for the non-goal vertex s_t. Let $a^* = \arg\min_{a\in A(s_t)} gd(succ(s_t,a))$. Then, $gd(s_t) = 1+gd(succ(s_t,a^*))$ and thus $s_t \neq succ(s_t,a^*)$. It follows that

$$\begin{aligned} u_{t+1}(s_t) &\leq u_{t+1}(succ(s_t,a^*)) + f(n) \\ &= u_t(succ(s_t,a^*)) + f(n) \\ &\leq f(n)gd(succ(s_t,a^*)) + f(n) \\ &= f(n)(1+gd(succ(s_t,a^*))) \\ &= f(n)gd(s_t). \end{aligned}$$

Furthermore, for all vertices $s \neq s_t$, it holds that $u_{t+1}(s) = u_t(s) \leq f(n)gd(s)$. Consequently, the induction hypothesis holds again at time step $t+1$.

2. Next, we prove by induction on the number of time steps t that $((f(n)+1)\sum_{s\in S} u_{t+1}(s) - u_{t+1}(s_{t+1})) - ((f(n)+1)\sum_{s\in S} u_t(s) - u_t(s_t)) \geq 1$, that is, that the potential $(f(n)+1)\sum_{s\in S} u(s) - u(s_t)$ increases by at least one with each movement. Consider an arbitrary time step t. The only u-value that changes is $u(s_t)$ for the non-goal vertex s_t. We distinguish three cases:

   (a) If $s_{t+1} \neq s_t$ (1) and $u_t(s_t) \leq u_t(s_{t+1})$ (2), then $u_{t+1}(s_{t+1}) = u_t(s_{t+1})$ according to (1). $u_{t+1}(s_t) \geq 1 + u_t(s_t)$ according to (2) and Condition 3. $u_t(s_t) - u_t(s_{t+1}) \geq -f(n)$ according to Condition 3. Consequently, the potential increases by

$$
\begin{aligned}
&((f(n)+1)\sum_{s\in S} u_{t+1}(s) - u_{t+1}(s_{t+1})) - ((f(n)+1)\sum_{s\in S} u_t(s) - u_t(s_t)) \\
&= ((f(n)+1)u_{t+1}(s_t) - u_{t+1}(s_{t+1})) - ((f(n)+1)u_t(s_t) - u_t(s_t)) \\
&\geq ((f(n)+1)(1+u_t(s_t)) - u_{t+1}(s_{t+1})) - f(n)u_t(s_t) \\
&= ((f(n)+1)(1+u_t(s_t)) - u_t(s_{t+1})) - f(n)u_t(s_t) \\
&= (f(n)+1) + (u_t(s_t) - u_t(s_{t+1})) \\
&\geq (f(n)+1) + (-f(n)) \\
&= 1.
\end{aligned}
$$

   (b) If $s_{t+1} \neq s_t$ (1) and $u_t(s_t) > u_t(s_{t+1})$ (2), then $u_{t+1}(s_{t+1}) = u_t(s_{t+1})$ according to (1). $u_t(s_{t+1}) \leq u_t(s_t) - 1$ according to (2) and Condition 1. $u_{t+1}(s_t) \geq u_t(s_t)$ according to Condition 2. Consequently, the potential increases by

$$
\begin{aligned}
&((f(n)+1)\sum_{s\in S} u_{t+1}(s) - u_{t+1}(s_{t+1})) - ((f(n)+1)\sum_{s\in S} u_t(s) - u_t(s_t)) \\
&= ((f(n)+1)u_{t+1}(s_t) - u_{t+1}(s_{t+1})) - ((f(n)+1)u_t(s_t) - u_t(s_t)) \\
&= ((f(n)+1)u_{t+1}(s_t) - u_t(s_{t+1})) - ((f(n)+1)u_t(s_t) - u_t(s_t)) \\
&\geq ((f(n)+1)u_{t+1}(s_t) - (u_t(s_t) - 1)) - ((f(n)+1)u_t(s_t) - u_t(s_t)) \\
&= (f(n)+1)(u_{t+1}(s_t) - u_t(s_t)) + 1 \\
&\geq 1.
\end{aligned}
$$

   (c) If $s_{t+1} = s_t$, then $u_{t+1}(s_t) \geq 1 + u_t(s_t)$ according to Condition 3 and the potential increases by

$$((f(n)+1)\sum_{s\in S} u_{t+1}(s) - u_{t+1}(s_{t+1})) - ((f(n)+1)\sum_{s\in S} u_t(s) - u_t(s_t))$$

$$
\begin{aligned}
&= ((f(n)+1)u_{t+1}(s_t) - u_{t+1}(s_{t+1})) - ((f(n)+1)u_t(s_t) - u_t(s_t)) \\
&= ((f(n)+1)u_{t+1}(s_t) - u_{t+1}(s_t)) - ((f(n)+1)u_t(s_t) - u_t(s_t)) \\
&= f(n)(u_{t+1}(s_t) - u_t(s_t)) \\
&\geq f(n)(1 + u_t(s_t) - u_t(s_t)) \\
&= f(n) \\
&\geq 1.
\end{aligned}
$$

3. Finally, we prove the main theorem. The potential $(f(n)+1)\sum_{s\in S} u(s) - u(s_t)$ is initially zero, increases with every movement by at least one, and is bounded from above by $(f(n)+1)\sum_{s\in S} f(n)gd(s)$. The main theorem follows.

□

Real-time search methods that satisfy Conditions 1, 2, and 3 of Theorem 14 cover (strongly connected) graphs repeatedly since the proof in Section 3 applies to them. In the following, we show how Theorem 14 can be used to prove that the cover times of single ants that use LRTA*, Wagner's Value-Update Rule, or Thrun's Value-Update Rule are polynomial in n. This is so because all three conditions of the theorem hold for these three real-time search methods for a function $f(n)$ that is polynomial in n. Then, $(f(n)+1)f(n)\sum_{s\in S} gd(s) \leq (f(n)+1)f(n)\sum_{i=0}^{n-1} i = (f(n)+1)f(n)(n(n-1)/2)$, according to our assumption that the graph is strongly connected. Consequently, the cover time is polynomial in n. On the other hand, all three conditions of the theorem hold for Node Counting only for a function $f(n)$ that can be exponential in n. For example, $f(n) \geq 2^{n/2-5/2}$ for the graph topology in Figure 7 since $u(g_1) - u(g_2) = 2^{n/2-5/2}$ after termination.

LRTA:* Condition 2 of Theorem 14 holds as shown by Theorem 2. Conditions 1 and 3 hold because, according to the value-update rule of LRTA*, $u_{t+1}(s_t) = 1 + \min_{a\in A(s_t)} u_t(succ(s_t, a))$, where $u_t(s)$ denotes the u-value of vertex s immediately before the ant updates the u-value of s_t and $u_{t+1}(s)$ the same u-value immediately after the update. That Condition 4 holds for $f(n) = 1$ was proved in [30]. Condition 4 holds on directed graphs for $f(n) = n - 1$.

Wagner's Value-Update Rule: Conditions 1, 2 and 3 of Theorem 14 hold because, according to Wagner's Value-Update Rule, $u_{t+1}(s_t) = 1 + u_t(s_t)$ if $u_t(s_t) \leq \min_{a\in A(s_t)} u_t(succ(s_t, a))$. That Condition 4 holds for $f(n) = 1$ was proved in [29]. Condition 4 holds on directed graphs for $f(n) = n - 1$.

Thrun's Value-Update Rule: Conditions 1, 2 and 3 of Theorem 14 hold because, according to Thrun's Value-Update Rule, $u_{t+1}(s_t) = \max(1 + u_t(s_t), 1 + \min_{a\in A(s_t)} u_t(succ(s_t, a)))$. That Condition 4 holds for $f(n) = n$ was proved in [27].

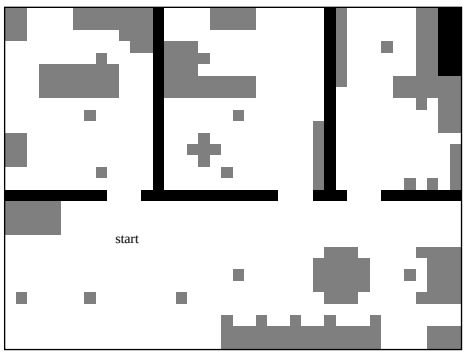


Figure 13. Terrain

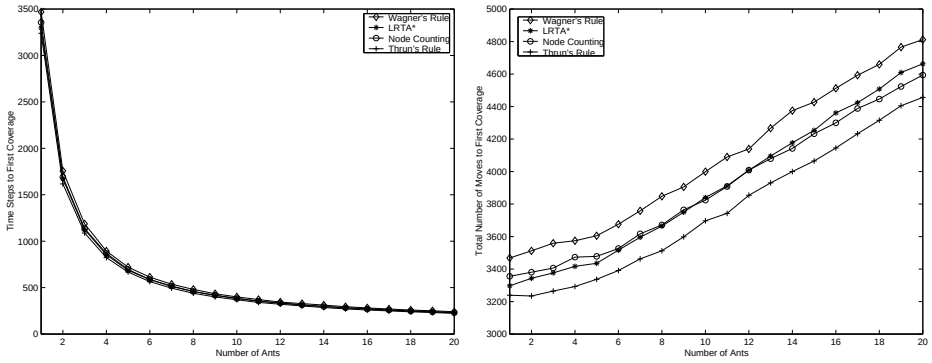


Figure 14. Cover Time and Total Number of Moves for First Coverage

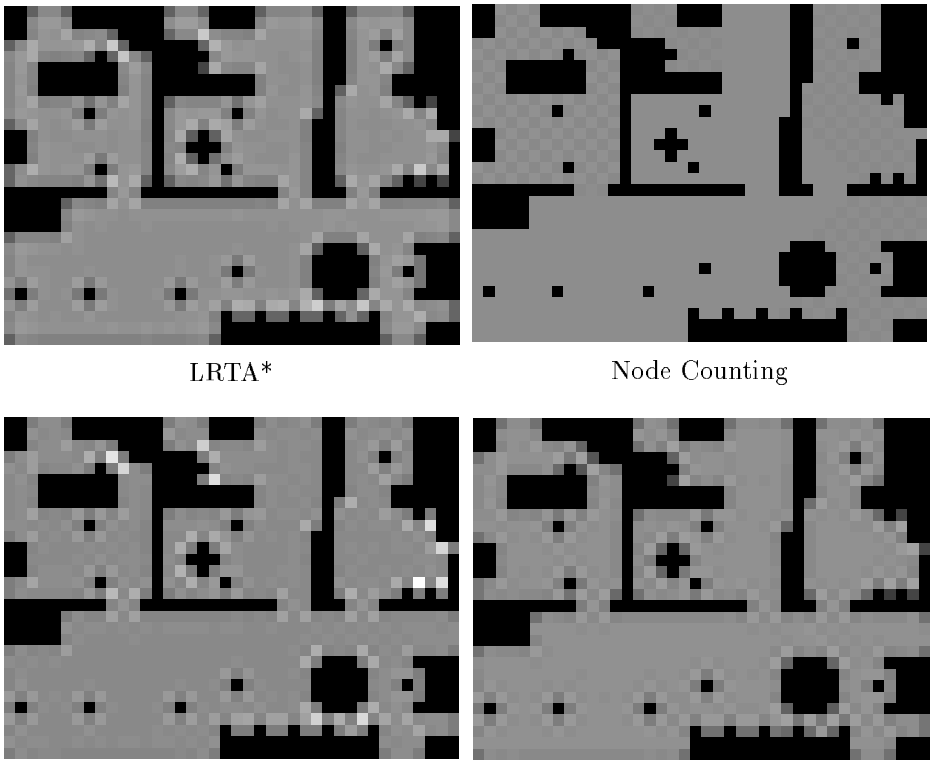
4.2. Cover Time: Experimental Results

In this section, we report experimental results for the cover times of vacuum-cleaning ants that use various real-time search methods, including LRTA* and Node Counting, in part of an office building that contains three offices and a small waiting area, see Figure 13 (the results for different terrains were similar). We imposed a regular four-connected grid over the terrain, resulting in 40×30 cells. The ants could move to each of the four neighboring cells of their current cell provided that the destination cell was traversable (white). Cells were untraversable if they contained either walls (black) or furniture (grey). All ants started in the same cell (marked "start") and used the same real-time search method, breaking ties randomly. The ants executed the real-time search method independently even if several ants shared a cell. Each ant moved once during each time step in a given sequential order.[2]

[2] We also performed experiments where the ants moved asynchronously or in random order but this led to similar results except that the variance of the total number of moves for the first coverage decreased with the number of ants instead of remaining roughly constant.

In our first three experiments, we studied the cover times of the real-time search methods for one-time vacuum cleaning and their visit frequencies for continuous vacuum cleaning. In our first experiment, we measured the number of time steps until the ants covered the terrain for the first time (cover time), averaged over 2,000 runs. This is important for one-time vacuum cleaning since each cell has to be vacuumed at least once. Figure 14 (left) shows the results. The trend was the same for all real-time search methods. The cover times improved as more ants were added although the rate of improvement decreased. In general, the cover times of all real-time search methods were similar. To differentiate better between the real-time search methods, Figure 14 (right) shows the total number of moves made by all ants, that is, (roughly) the number of ants times the cover time. It appears that the difference in the total number of moves of the different real-time search methods was independent of the number of ants. Thrun's Value-Update Rule was best, followed by LRTA* and Node Counting (that were almost indistinguishable), and finally followed by Wagner's Value-Update Rule. The cover time of single ants was between 3,200 and 3,500 with a standard deviation between 600 and 700. Thus, the difference in performance was dominated by the standard deviation. This experimental result is interesting because our theoretical worst-case results suggested that the cover time of Node Counting might be much larger than the cover time of LRTA*. We also experimented with random walks but their cover time was much larger than those of the real-time search methods above. For example, the cover time of single ants that use random walks was approximately 49,000 with a standard deviation of about 20,000.

In our second experiment, we measured the frequency with which the ants visited each cell of the terrain when they covered the terrain repeatedly. This is important for continuous vacuum cleaning because one probably wants to vacuum each cell equally often in the long run (although one could argue that most dirt accumulates where people walk and thus that one wants to vacuum cells close to obstacles less often). We stopped the ants after 2,000,000 time steps. The visit frequencies did not change significantly as the number of ants increased, no matter which real-time search methods were used. We therefore report the visit frequencies for single ants only. Figure 15 shows the results. Darker cells in the figure were visited less frequently. As the figure shows, some real-time search methods visited the cells more uniformly than others. For example, Node Counting visited the cells much more uniformly than LRTA*. This is interesting because the total number of movements of LRTA* and Node Counting was similar in our first experiment. One measure for the uniformity of the visit frequencies is their entropy. The larger the entropy $-\sum_s P(s) \log_2 P(s)$, the closer to uniform the visit frequencies $P(s)$ are. The entropy of uniform visit frequencies is 9.7830 and the entropies of the visit frequencies of the real-time search methods were: Node Counting (9.7829), Wagner's Value-Update Rule (9.7779), Thrun's Value-Update Rule (9.7772), and LRTA* (9.7727). Another measure for the uniformity



LRTA* Node Counting

Wagner's Value-Update Rule Thrun's Value-Update Rule

Figure 15. Visit Frequencies for Repeated Coverage

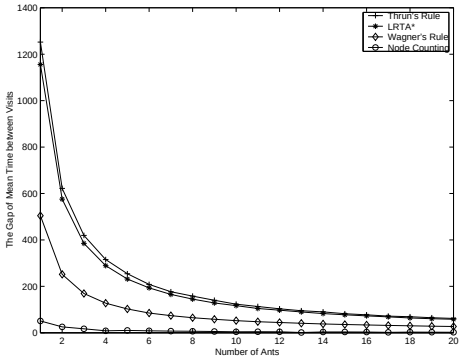


Figure 16. Difference of Largest and Smallest Mean Time between Visits for Repeated Coverage

of the visit frequencies is the difference between the mean times between visits to the cells with the largest and smallest mean time between visits. The smaller this value, the closer to uniform the visit frequencies. Figure 16 shows the results, that were similar to those for the entropies. Only the positions of Thrun's Value-

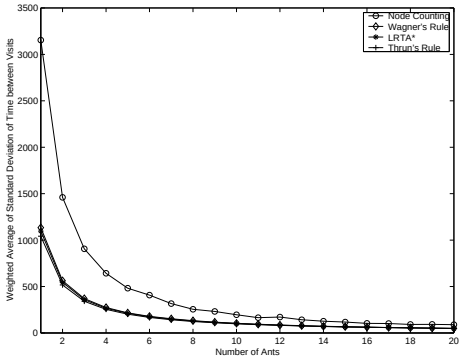


Figure 17. Standard Deviation of the Time between Visits for Repeated Coverage

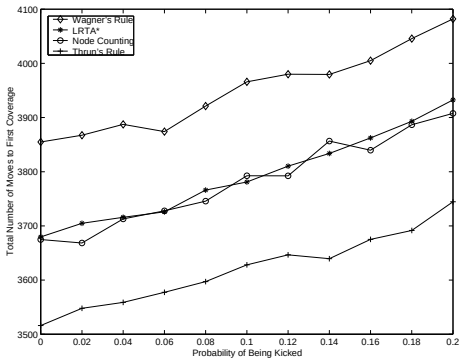


Figure 18. Total Number of Moves for First Coverage if the Ants Were Kicked

Update Rule and LRTA* were switched.

In our third experiment, we measured whether the times between visits to the cells were spread out evenly. This is important for continuous vacuum cleaning because it seems better if a cell gets visited every 100 time steps than if it gets visited every 10 time steps for 9 times in a row and then again only after 910 time steps. One measure of how evenly the times between visits are spread out is the average of the standard deviations of the times between visits over all cells, weighted with their visit frequencies. The closer to zero this value, the more evenly spread out the times between visits. We stopped the ants after 2,000,000 time steps. Figure 17 shows the results. The times between visits were very unevenly spread out for Node Counting and more evenly spread out for all other real-time search methods, with almost no difference among them. This is interesting because Node Counting had very uniform visit frequencies in our second experiment.

In the next three experiments, we studied the cover times of the real-time search methods for one-time vacuum cleaning under various failure conditions. (We also performed experiments where we studied the visit frequencies of the

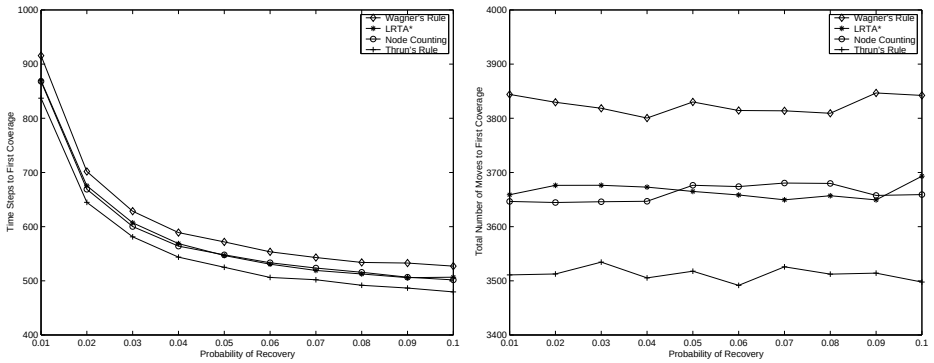


Figure 19. Cover Time and Total Number of Moves for First Coverage if Ants Malfunction (recover probability varies)

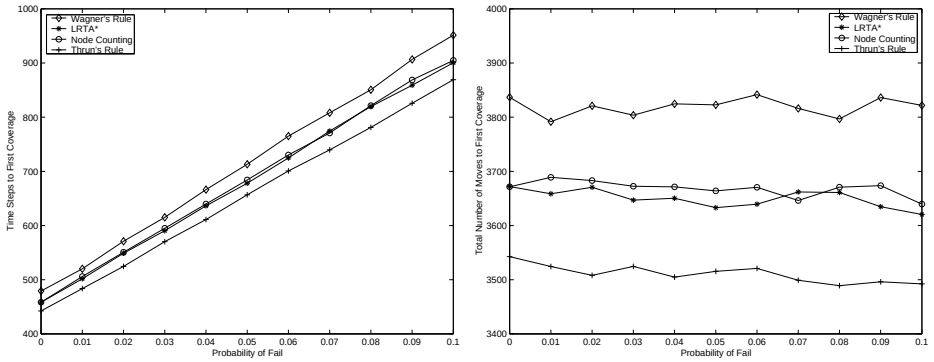


Figure 20. Cover Time and Total Number of Moves for First Coverage if Ants Malfunction (failure probability varies)

real-time search methods for continuous vacuum cleaning but the effect of the failure conditions on the visit frequencies was small.) In our fourth experiment, we measured the robustness of the real-time search methods when the ants were moved away from their current cell without realizing this. This is important because people can easily run into small vacuum-cleaning ants and accidentally push them to a different location. During each time step, with a given probability exactly one ant was moved (otherwise no ant was moved). If an ant was moved, exactly one ant and its new cell were chosen with uniform probability, with the restriction that the ant was moved by at most two cells. Figure 18 shows the total number of moves until eight ants covered the terrain for the first time, averaged over 2,000 runs. All real-time search methods continued to cover the terrain, and the number of movements increased gracefully as the probability of being moved increased. The order of the real-time search methods remained unaffected by this failure condition.

In our fifth experiment, we measured the robustness of the real-time search

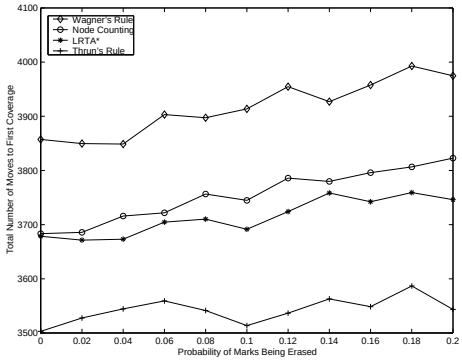


Figure 21. Total Number of Moves for First Coverage if Markings Were Erased

methods when the ants failed. This is important because robots can malfunction. During each time step, each functional ant failed with a given failure probability and each failed ant can recovered with a given recovery probability. Figure 19 shows the cover time (left) and total number of moves (right) until eight ants covered the terrain for the first time, averaged over 2,000 runs. We kept the failure probability constant at 0.01 and varied the recovery probability from 0.01 to 0.10 (implying that it took a failed ant an average of 10 to 100 time steps to recover). Figure 20 shows the results of a similar experiment where we kept the recovery probability constant at 0.10 and varied the failure probability from 0.00 to 0.10. All real-time search methods continued to cover the terrain as long as the failure probability was no larger than the recovery probability. The number of movements remained roughly the same and the cover time increased gracefully as the failure probability increased or the recovery probability decreased. The order of the real-time search methods remained unaffected by this failure condition.

In our sixth and last experiment, we measured the robustness of the real-time search methods when the markings were erased. This is important because physical markings can get destroyed. During each time step, with a given probability exactly one marking was erased (otherwise no marking was erased). If a marking was erased, exactly one cell was chosen with uniform probability and its u-value was set to zero (because ants cannot distinguish between a cell without a marking and a cell whose marking was destroyed). Figure 21 shows the total number of moves until eight ants covered the terrain for the first time, averaged over 2,000 runs. All real-time search methods continued to cover the terrain, and the number of movements increased gracefully as the probability with which markings were erased increased. The order of the real-time search methods remained unaffected by this failure condition.

5. Conclusion

We studied real-time search methods that allow ants to cover terrain, as required for one-time or continuous vacuum cleaning, lawn mowing, mine sweeping, or surveillance. In particular, we studied LRTA* and Node Counting, two real-time search methods that are algorithmically similar. Node Counting, for example, always moves the ant to a neighboring location that has been visited the least number of times. Our experimental results on regular four-connected grids indicated that they result in similar cover times. Therefore, it could be speculated that ants that use Node Counting guarantee a similar cover time as ants that use LRTA*. We showed that the cover time of ants that use LRTA* is guaranteed to be polynomial in the number of locations of the terrain. Then, we showed, surprisingly, that the cover time of ants that use Node Counting can be exponential in (the square root of) the number of locations even in terrain that corresponds to (planar) undirected trees. Thus, while ants that use Node Counting experimentally performed well in some terrains that correspond to planar undirected graphs, this property alone is not sufficient to explain why they performed well since their cover time on planar undirected graphs is not always good. This means that one is not assured that ants that use Node Counting exhibit a good cover time in terrains whose topology differs from the topology of the terrains used for testing. On the other hand, ants that use LRTA* always guarantee a good cover time, no matter whether the terrains correspond to directed graphs or whether they correspond to planar graphs. The cover time guarantee of ants that use LRTA* does not seem to come at the cost of a worse average-case cover time in practice since ants that use LRTA* seem to perform as well as ants that use Node Counting on regular four-connected grids. Other advantages of LRTA* over Node Counting have been studied in [15].

Our results demonstrate that theoretical results are important because it is unlikely that one will find worst-case example graphs by randomly generating graphs and experimentally determining how long it takes ants that use a given real-time search method to cover them. Our results also demonstrate that applications of real-time search methods are important because they suggest which kinds of terrains are important in practice and thus should be analyzed theoretically. For example, our worst-case example graphs might not occur in practice and thus ants that use real-time search methods might be able to guarantee a better cover time in practice than is suggested by a worst case analysis over all terrain topologies. Consequently, our results raise a variety of questions and thus suggest future work that follows from the discrepancy of the theoretical worst-case results and the experimental average-case results on grids. For example, it would be interesting to perform a theoretical average-case analysis (say, for random tie breaking). Even in the context of a theoretical worst-case analysis, it would be interesting to identify realistic properties of graphs that guarantee ants that use Node Counting a cover time that is polynomial in the number of vertices.

Grids, for example, are undirected graphs whose vertices have a bounded degree, and no worst-case results are known for graphs with this property.

Acknowledgements

The Intelligent Decision-Making Group at Georgia Tech is supported by an NSF Career Award to Sven Koenig under contract IIS-9984827. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations and agencies or the U.S. government.

References

[1] F. Adler and D. Gordon. Information collection and spread by networks of patrolling ants. *The American Naturalist*, 140(3):373–400, 1992.

[2] T. Balch and R. Arkin. Avoiding the past: A simple, but effective strategy for reactive navigation. In *International Conference on Robotics and Automation*, pages 678–685, 1993.

[3] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 73(1):81–138, 1995.

[4] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, pages 52–61, 2000.

[5] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism. In *Proceedings of the National Conference on Artificial Intelligence*, pages 714–719, 1997.

[6] R. Brooks and A. Flynn. Fast, cheap, and out of control: A robot invasion of the solar system. *Journal of the British Interplanetary Society*, pages 478–485, 1989.

[7] G. Chartrand and L. Lesniak. *Graphs and Digraphs*. Wadsworth and Brooks/Cole, second edition, 1986.

[8] T. Ishida. *Real-Time Search for Learning Autonomous Agents*. Kluwer Academic Publishers, 1997.

[9] T. Ishida and R. Korf. Moving target search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 204–210, 1991.

[10] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[11] S. Koenig, A. Blum, T. Ishida, and R. Korf, editors. *Proceedings of the AAAI-97 Workshop on On-Line Search*. AAAI Press, 1997. Available as AAAI Technical Report WS-97-10.

[12] S. Koenig and R.G. Simmons. Real-time search in non-deterministic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1660–1667, 1995.

[13] S. Koenig and R.G. Simmons. Easy and hard testbeds for real-time search algorithms. In *Proceedings of the National Conference on Artificial Intelligence*, pages 279–285, 1996.

[14] S. Koenig and R.G. Simmons. Solving robot navigation problems with initial pose uncertainty using real-time heuristic search. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 145–153, 1998.

[15] S. Koenig and B. Szymanski. Value-update rules for real-time search. In *Proceedings of the National Conference on Artificial Intelligence*, pages 718–724, 1999.

[16] R. Korf. Real-time heuristic search: First results. In *Proceedings of the National Conference on Artificial Intelligence*, pages 133–138, 1987.

[17] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

[18] N. Nilsson. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, 1971.

[19] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1985.

[20] A. Pirzadeh and W. Snyder. A unified solution to coverage and search in explored and unexplored terrains using indirect control. In *Proceedings of the International Conference on Robotics and Automation*, pages 2113–2119, 1990.

[21] R. Russell. Heat trails as short-lived navigational markers for mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 3534–3539, 1997.

[22] R. Russell, D. Thiel, and A. Mackay-Sim. Sensing odour trails for mobile robot navigation. In *Proceedings of the International Conference on Robotics and Automation*, pages 2672–2677, 1994.

[23] S. Russell and E. Wefald. *Do the Right Thing – Studies in Limited Rationality*. MIT Press, 1991.

[24] R. Sharpe and B. Webb. Simulated and situated models of chemical trail following in ants. In *Proceedings of the International Conference on Simulation of Adaptive Behavior*, pages 195–204, 1998.

[25] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[26] B. Szymanski and S. Koenig. The complexity of node counting on undirected graphs. Technical report, Computer Science Department, Rensselaer Polytechnic Institute, Troy (New York), 1998.

[27] S. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, School of Computer Science, Carnegie Mellon University, Pittsburgh (Pennsylvania), 1992.

[28] S. Thrun. The role of exploration in learning control. In D. White and D. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pages 527–559. Van Nostrand Reinhold, 1992.

[29] I. Wagner, M. Lindenbaum, and A. Bruckstein. On-line graph searching by a smell-oriented vertex process. In S. Koenig, A. Blum, T. Ishida, and R. Korf, editors, *Proceedings of the AAAI Workshop on On-Line Search*, pages 122–125, 1997. Available as AAAI Technical Report WS-97-10.

[30] I. Wagner, M. Lindenbaum, and A. Bruckstein. Efficiently searching a dynamic graph by a smell-oriented vertex process. *Annals of Mathematics and Artificial Intelligence*, 24:211–223, 1998.