

Collaborative Foraging using Beacons

Brian Hrolenok, Sean Luke, Keith Sullivan, and Christopher Vo
Department of Computer Science, George Mason University
MSN 4A5, Fairfax, VA 22030, USA
{bhroleno, sean, ksulliv2, cvo1} @ cs.gmu.edu

ABSTRACT

A classic example of multiagent coordination in a shared environment involves the use of pheromone deposits as a communication mechanism. Due to physical limitations in deploying actual pheromones, we propose a sparse representation of the pheromones using movable beacons. There is no communication between the beacons to propagate pheromones; instead, robots make movement and update decisions based entirely on local pheromone values. Robots deploy the beacons throughout the environment, and subsequently move them and update them using a variation of value iteration. Simulation results show that our approach is effective at finding good trails, locally improving them, and adapting to dynamic changes in the environments.

Categories and Subject Descriptors

I.2 [Distributed Artificial Intelligence]: Multiagent Systems

General Terms

Algorithms

Keywords

Beacon, Ant, Pheromone

1. INTRODUCTION

One challenge in swarm robotics is performing effective communication. Broadcast communication may be unattractive due to environmental factors, limited range and power, lack of global communications infrastructure, or overly congested channels. Instead, swarm robotics research has often focused on local interaction and forms of indirect communication such as pheromone deposits. In this paper we show a formal approach to pheromone deposit communication via beacons deployed in the environment by the robots. (We will use the terms “robot” and “ant” interchangeably.)

In previous work [16] we presented a technique to enable a large swarm of ant-like agents to perform foraging and other trail-following tasks by depositing, updating, and following pheromones in the environment. Unlike much previous work in the multiagent pheromone-based foraging literature, our approach cast pheromones as state utility values, and applied formal utility update

equations based loosely on value iteration and temporal differencing. The work also employed multiple simultaneous pheromones to direct various aspects of the task at hand. Using these equations, we were able to demonstrate collaborative foraging using two pheromones (one which ultimately defined a gradient to food, and one back to the ants’ nest). We went further than this, demonstrating trail optimization, discovery of self-intersecting tours involving many waypoints (and many pheromones), and adaptation to dynamic changes in the environment.

That work was motivated in part by our research in swarm robotics: we sought a formal language and substrate with which swarms of agents might perform a range of cooperative tasks. The approach was also of value to swarm robotics because it did not require odometry or localization, and had a light computational load. However, the agents lived in a 2d discretized grid-world environment, with pheromones stored in the grid of the environment itself.

The present paper represents a significant advance over this early work, adapting the pheromone idea to a more realistic robot scenario through the use of deployable and modifiable beacons as a sparse representation of the pheromone environment. The ant robots deploy the beacons, and subsequently follow, update, move, and remove them. From the perspective of the robots, the beacons are a graph of states with utility values, just as the earlier grid-world was a denser graph of the same. Each beacon stores one value per pheromone employed by the swarm. Beacons only represent local information, and do not communicate with one another to spread pheromones. Rather, after deploying the beacons, robots update pheromone values using similar equations as [16], and likewise the robots make routing decisions based on the pheromones in nearby beacons. After discovering a trail, the robots may additionally move the beacons in order to optimize it. The work presented uses robots in simulation, but its approach is specifically designed to be deployable to actual robots in our laboratory. Our purpose here is to extend the approach taken in [16] to environments with more realistic assumptions, taking a significant step towards deploying to physical robots in our laboratory.

We begin the paper with a discussion of previous work in pheromone-based multiagent interaction, marker-based robotic navigation and optimization, and other related topics. We then introduce and discuss the proposed model, followed by experimental work with the model.

2. PREVIOUS WORK

Indirect communication through pheromone deposits is an example of *stigmergy*, a term coined by Pierre-Paul Grassé in the 1950s to describe a mechanism in which colonies of a certain genus of termites collaborate to build a nest [4, 7]. Many examples of stigmergy appear in nature, including the foraging behavior of ants.

Cite as: Collaborative Foraging using Beacons, Hrolenok, Luke, Sullivan, and Vo, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX. Copyright © 2010, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Ants leave pheromone deposits as they move in the environment, and their navigation is stimulated through local observations of pheromone strength and gradient. The global ant foraging behavior emerges through these simple local pheromone interactions. Other familiar kinds of indirect communication through the environment include leaving footsteps in the snow; or leaving trails of pebbles or breadcrumbs along a journey in order to help find a way home.

The collective behavior of ants that emerges from communication through pheromones has been widely studied in both artificial life and in robotics. Beyond providing a robust, decentralized, and distributed means of communication, pheromone-based swarms have also shown the ability to optimize trails, as observed in [6]. The networks of paths constructed by ants have been compared to minimal spanning trees [18] which may emerge from the rapid accumulation of pheromone strength across shorter paths [2, 3]. By incorporating stochastic exploration and evaporation of old pheromones, swarms can be shown to adapt to dynamic situations such as changing goals or obstacles [12, 20]. We have demonstrated several of our own examples in [16].

Some pheromone-based reinforcement learning algorithms have been proposed for foraging problem domains. Most of these involve agents that use a fixed pheromone depositing procedure, and incorporate the existing pheromone values into their action selection and update mechanisms [8, 10, 11].

Several works have explored agent behaviors that take advantage of larger vocabularies of pheromones. For example, multiple-pheromone models have been used to establish separate gradients for different tasks [5, 16, 25], rather than relying on arbitrary *a priori* mechanisms to augment a one-pheromone model. Vaughan et al. have proposed an alternative involving pheromones that additionally indicate direction. Parunak et al. [17, 18] have exploited a variety of techniques such as using pheromones with different semantics, pheromones with different dynamics (e.g. different rates of evaporation and propagation) and using history (e.g. weighing pheromones more strongly in the direction of motion).

Several of the ant-inspired methods described so far rely on the ability of agents to modify the environment. While there has been some work on a robotic mechanism to deposit and sense chemical trails [21], this could be impractical in many real scenarios. One common approach is to rely on existing communications mechanisms to share internal pheromone models. For example, in [22, 23, 24], Vaughan et al. simulated stigmergic communication by making it possible for agents to share trails of waypoints over a wireless network. Similarly, Payton et al. experimented with “virtual pheromones” based on propagating discrete messages between mobile robots with short-range omni-directional infrared communications [19]. O’Hara et al. introduced the GNATs, a pre-deployed line-of-sight sensor network to support robotic navigation tasks [15]. They demonstrated successful distributed path planning in this infrastructure using variants of the wavefront planning algorithm [13] and the distributed Bellman-Ford algorithm [14]. Barth et al. proposed a dynamic programming approach to swarm navigation based on deployable, but immobile relay markers [1]. This approach also relies on communication between the markers in order to establish and update the pheromone values. Ziparo et al. used deployable, non-mobile RFID tags to help path planning and exploration with large groups of robots [26].

It is important to note that the goal of our work is not the analysis of the network structure created by the topology of the beacons in the environment, as this has been well covered by other researchers. Instead, we focus on swarm robot behaviors which may successfully employ a collective model in the face of severe communications constraints (in this case, for the foraging task).

3. MODEL

Our foraging world is a bounded two-dimensional continuous environment. A *nest* is located somewhere in the environment, holding some number of robots (we will refer to them simply as *ants*). The environment also holds some N *food* source locations (in our experiments, we set $N = 1$). The environment also may contain obstacles through which the ants may not travel.

The ants’ task is to bring as much food back to the nest as possible within a given time frame. This consists of several subtasks. First, the ants must find a remote food location, and establish a trail there. Second, the ants must repeatedly follow the trail to the food location, harvest some food, ferry it back along the trail to the nest, and deposit it there. Third, the ants should optimize the trail, globally adopting new routes when they are found to be shorter, or attempting to straighten (and shorten) the current trail locally.

To assist them in this task, the ants deposit three pheromones and read them at various beacons the ants have deployed and organized throughout the environment. Each beacon represents the state of pheromones in the environment at that location. The three pheromones are:

- The *foraging pheromone*, used to build a gradient to food.
- The *ferrying pheromone*, used to build a gradient to the nest.
- The *wandering pheromone*, used to indicate the how often a state has been visited.

Beacons and ants have the same communications *range*. Beacons within the range of one another, and not occluded by an obstacle, are defined as *neighbors*. Each ant will associate itself with the nearest non-occluded beacon within its range, if any, and this beacon is called the ant’s *current beacon*. Ants can read and update pheromone information stored in their current beacon, and (notionally by temporarily moving to the current beacon) its immediate neighbors. Ants can also detect if food or the nest are within range.

Ants do not directly communicate with each other, and likewise beacons do not communicate with each other. In fact, beacons need not be wireless or even active: they could be buckets, RFID tags, or other markers with limited visibility range. Ants only need to be able to identify and home in on individual beacons, and to store and retrieve data in them.

Each ant has a *mode*, either FORAGING or FERRYING, indicating the ant’s current task. All ants start FORAGING and located at the nest. At each iteration an ant updates the pheromones of its current beacon (if any), then performs one action (such as moving to the food or nest; exploring randomly; moving, deploying, or deleting a beacon; etc.), then the ant updates the pheromones again for good measure. The action decision process is described later. Afterwards, each beacon *depletes* (evaporates) its *foraging* and *ferrying* pheromone values by multiplying them by a fixed constant $0 < \beta \leq 1$. Except in the case of moving obstacles, depletion is not required (β can be 1). We set $\beta = 0.9$. The wander pheromone is not depleted; it is updated as described below.

3.1 States and Pheromones

Ants spread pheromone values from beacon to beacon using roughly the same method as in the (beacon-less) [16]: a form of value iteration. From a dynamic programming perspective, beacons are the *states* in the environment, and pheromone values are the *utilities* U_p of those states, one per pheromone p . To illustrate state (beacon) transitions, consider Figure 1, where the ant has recently transitioned from state (beacon) s to state s' and may transition to any of the s''_i states in the immediate future. After reaching a

goal via a series of transitions, an ant will receive a positive reward R_p and change its mode from FORAGING to FERRYING or vice versa. Specifically, when the ant has just changed to (or is starting as) FORAGING, $R_{ferrying}$ is set to a positive constant REWARD, and when the ant has just changed to FERRYING, $R_{foraging}$ is set to REWARD. In all other cases, $R_p = 0$ for all p .

Updating. To be maximally general, let us start with the assumption that for each s''_i , the ants have available a model $T(s', a, s''_i)$ indicating the probability of transitioning to s''_i from s' if the ant chooses to perform some action $a \in A$. The update rule for each pheromone $p \in \{foraging, ferrying\}$ is a variation of the Bellman Equation in which U_p does not decrease:

$$U_p(s') \leftarrow \max \left(U_p(s'), R_p + \gamma \max_{a \in A} \sum_i T(s', a, s''_i) U_p(s''_i) \right)$$

where γ is a value between 0 and 1. However, in actuality transitions are deterministic (or at any rate, T is not available to the ants), which reduces the update equation to:

$$U_p(s') \leftarrow \max \left(U_p(s'), R_p(s') + \gamma \max_i U_p(s''_i) \right) \quad (1)$$

The *wandering pheromone* is updated so that the ant will move away from commonly-visited beacons. Specifically:

$$U_{wandering}(s') \leftarrow U_{wandering}(s') - 1 \quad (2)$$

Following. When it is in the mode of transitioning from beacon to beacon, an ant's policy is fixed: if the ant is FORAGING, or FERRYING, or in certain situations *wandering*, it will move to the neighboring state beacon whose pheromone for that mode is highest. That is, it will select the beacon

$$\operatorname{argmax}_i U_p(s''_i)$$

where p is *foraging*, *ferrying*, or *wandering*, as appropriate. We refer to this action as *following* the pheromone trail.

It's important to note here that the ants are updating *all* the pheromones, but are only transitioning along *one* pheromone at a time. This has an important quality which crucially differentiates this method from plain value iteration. Because the ant is updating all pheromones as it transitions from the food to the nest (or back), it can build up a gradient in $O(n)$ time, where n is the number of transitions. Contrast this to traditional dynamic programming or reinforcement learning approaches, whose repeated backups require $O(n^2)$ time. This significant improvement in complexity is made possible by the symmetry of the environment in the ant foraging problem: the probability of transitioning from state s_i to state s_j is the same as the probability of transitioning from s_j to s_i .

3.2 Algorithm with Pre-Positioned Beacons

To introduce the algorithm more gently, we first present a version that considers the beacons as fixed in the environment. This algorithm is the sparse analogue of the dense pheromone grid structure in [16] and is shown here to demonstrate its close similarity. In the next section we will then extend this to the full algorithm, with beacon deployment, movement, and deletion.

Robustness can be accomplished with some degree of randomness. We add this in two ways. First, with small probability the ant will initiate a temporary *exploration mode* in which it performs random transitions some number of times before resuming normal

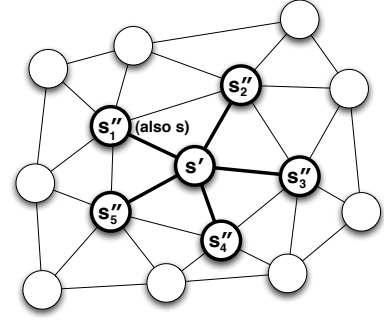


Figure 1: State transition example for the beacon model. The ant has just transitioned from state s to state s' , and now is preparing to transition to one of the s'' . State transitions are reflexive.

operation. Second, if the ant does not (with some probability) decide to do its standard “pheromone following” procedure, it will instead follow the *wandering* pheromone, which pushes it to areas not well explored.

The algorithm relies on certain constants: REWARD (set to 1.0) is how much reward the ant receives when it reaches the nest or food; COUNT is how long the ant will stay in exploration mode; $p_{Explore}$ is how likely the ant will enter exploration mode; and p_{Follow} is how likely the ant will do pheromone following. An ant will always have a current beacon unless *no* beacons are yet placed near the nest. The algorithm looks like this:

```

1: global variables:
2:    $mode \leftarrow$  FORAGING,  $count \leftarrow 0$ , and  $reward \leftarrow$  REWARD
3: loop
4:    $c \leftarrow$  compute current beacon, if any
5:   if  $c$  exists then
6:     UpdatePheromones( $c$ )
7:   if food within range of me and  $mode=$ FORAGING then
8:     Move to food,  $mode \leftarrow$  FERRYING,
9:      $reward \leftarrow$  REWARD
10:  else if nest within range of me and  $mode=$ FERRYING then
11:    Move to nest,  $mode \leftarrow$  FORAGING,
12:     $reward \leftarrow$  REWARD
13:  else if  $count > 0$  and  $c$  exists and has neighbors then
14:    Move to random neighbor of  $c$ ,  $count = count - 1$ 
15:  else if  $\text{Rand}(p_{Explore})$  then
16:     $count \leftarrow$  COUNT
17:  else if  $c$  exists and  $\text{CanFollow}(c)$  and  $\text{Rand}(p_{Follow})$  then
18:    Follow( $mode, c$ )
19:  else if  $c$  exists then
20:    Follow(WANDERING,  $c$ )
21:   $c \leftarrow$  recompute current beacon, if any
22:  if  $c$  exists then
23:    UpdatePheromones( $c$ )

```

This algorithm directly extends [16] to the more general connectivity graph case. The functions used in the algorithm are:

UpdatePheromones(current beacon). Each pheromone is updated using Equations 1 and 2, where s' is the current beacon, and where either $R_{foraging}$ or $R_{ferrying}$ is set to the current *reward*, depending on whether the *mode* is presently FERRYING or FORAGING respectively. The other R_p is set to 0.

Rand(prob). Return *true* with probability *prob*, else *false*.

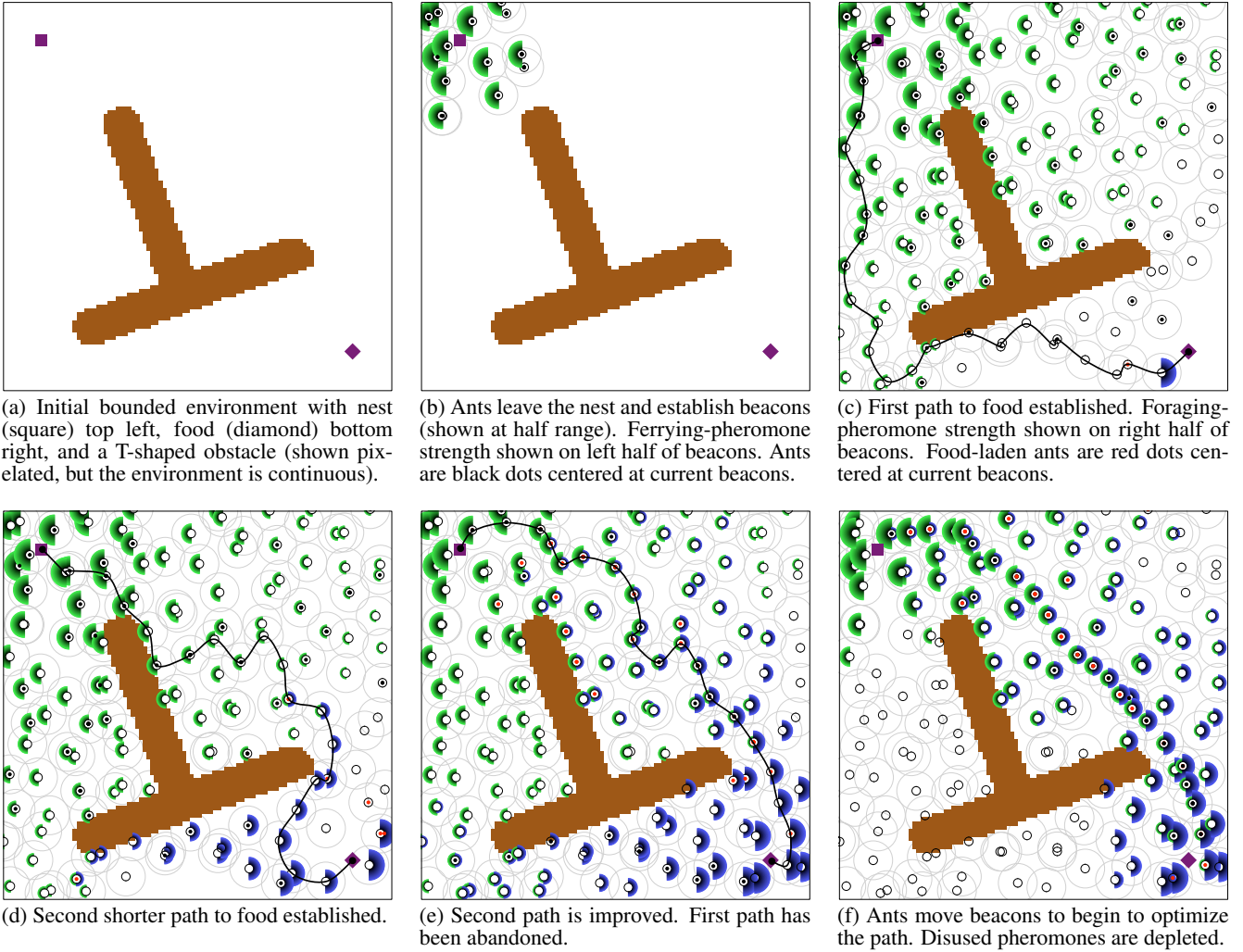


Figure 2: Example trace of the algorithm in action.

CanFollow(mode, current beacon). Let beacon B be the neighbor of the current beacon with the highest pheromone value corresponding to *mode*. If B exists and its pheromone value is > 0 , return *true*, else *false*.

Follow(mode, current beacon). Move to the neighbor of the current beacon with the highest pheromone value corresponding to *mode* (break ties randomly).

3.3 Deploying, Moving, and Deleting Beacons

This initial algorithm is sensitive to beacon location. If the beacons are positioned poorly, the ant trail will be suboptimal; and if the graph is disconnected, the ants may not be able to find a trail to food at all. For these reasons it is advantageous for the ants to be able to *deploy* the beacons on their own, then later *move* them and ultimately *remove* excess or unnecessary beacons to optimize the graph. We now extend the algorithm to include these cases.

This requires a few new constants: p_{Deploy} is the probability of deploying a new beacon, and p_{Move} is the probability of moving the current beacon. Beacon deletion always occurs if it is feasible. Certain other constants are described later. The algorithm only differs from the previous one in certain lines, denoted with $\lceil \rfloor$.

The revised algorithm is:

```

1: global variables:
2:    $mode \leftarrow \text{FORAGING}$ ,  $count \leftarrow 0$ , and  $reward \leftarrow \text{REWARD}$ 
3: loop
4:    $c \leftarrow$  compute current beacon, if any
5:   if  $c$  exists then
6:     UpdatePheromones( $c$ )
7:   if food within range of me and  $mode = \text{FORAGING}$  then
8:     Move to food,  $mode \leftarrow \text{FERRYING}$ ,
9:      $reward \leftarrow \text{REWARD}$ 
10:  else if nest within range of me and  $mode = \text{FERRYING}$  then
11:    Move to nest,  $mode \leftarrow \text{FORAGING}$ ,
12:     $reward \leftarrow \text{REWARD}$ 
13:  else if  $c$  exists and CanRemove( $c$ ) then ┌
14:    Remove( $c$ ) └
15:  else if  $count > 0$  and  $c$  exists and has neighbors then
16:    Move to random neighbor of  $c$ ,  $count = count - 1$ 
17:  else if Rand( $p_{Explore}$ ) then
18:     $count \leftarrow \text{COUNT}$ 
19:  else if  $c$  exists and CanMove( $c$ ) and Rand( $p_{Move}$ ) then ┌
20:    Move( $c$ ) └
21:  else if  $c$  exists and CanFollow( $c$ ) and Rand( $p_{Follow}$ ) then

```

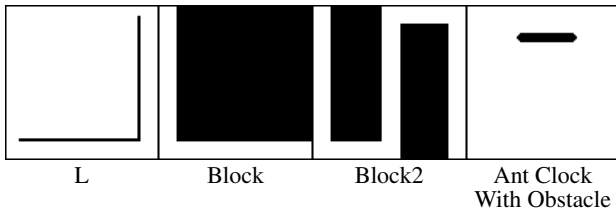


Figure 3: Four experimental obstacle environments. Left to right: L, Block, Block2, Ant Clock Obstacle. White is free space and black is the obstacle.

```

22:     Follow(mode,c)
23: [   else if CanDeploy() and Rand( $p_{Deploy}$ ) then           ]
24: [     Deploy()                                           ]
25:   else if  $c$  exists then
26:     Follow(WANDERING,c)
27: [   else move to the closest beacon, breaking ties randomly ]
28:    $c \leftarrow$  recompute current beacon, if any
29:   if  $c$  exists then
30:     UpdatePheromones(c)

```

Note line 27: deleting or moving a beacon can cause an ant associated with that beacon to become stranded such that there are *no* beacons within its range. On line 27 the ant searches (for example, in a spiral) to find and move to the closest beacon.

The new deployment, deletion, and movement functions are:

CanDeploy(). The goal is to only deploy a beacon into an uncrowded region, and only if there are beacons left to deploy.

This requires three new constants: DEPLOYTRIES (10), DEPLOYRANGE (0.9), and DEPLOYCROWD (0.6). If the maximum number of beacons has been reached, return *false*. Otherwise the ant tries DEPLOYTRIES times to find a random non-occluded location no further than DEPLOYRANGE \times range away from the current beacon, or from the ant (if there is no current beacon), such that there is no beacon within DEPLOYCROWD \times range of that location. If a location was found return *true*. Else return *false*.

The maximum number of beacons controls the overall beacon density. Since we are most interested in sparse beacon deployment, we've set this low (3).

Deploy(). Deploy a new beacon at the location computed by CanDeploy(). Move the ant to that location.

CanMove(current beacon). The goal is to move the beacon precisely in-between neighbors likely to be on the ant trail, so as to straighten the trail, without breaking other possibly important trails.

Let locations B_1 and B_2 be the positions of the neighbors of the current beacon with the highest *foraging* and *ferrying* pheromones respectively, breaking ties randomly; let P_1 and P_2 be those *foraging* and *ferrying* pheromone values; and let W be the minimum *wander* pheromone of the two. If the food is within range of the ant, replace B_1 with the food location and set $P_1 = \text{REWARD}$; likewise if the nest is within the range of the ant, replace B_2 with the nest location and set $P_2 = \text{REWARD}$.

Compute a new location that is the midpoint between B_1 and B_2 . Return *false* if any of the following are true:

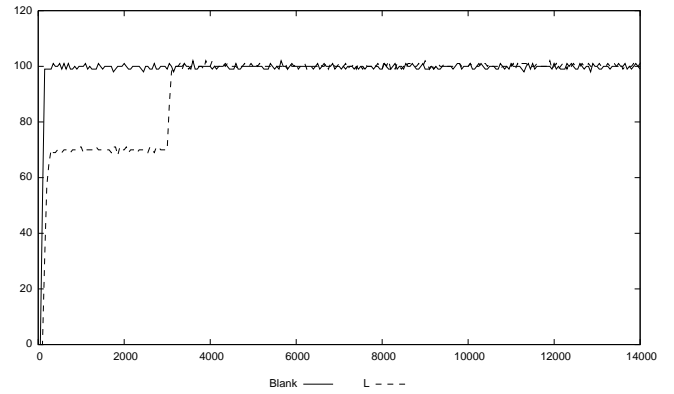


Figure 4: Mean food collected for the L environment with $p_{Explore} = 0.1, p_{Move} = 0$

- If B_1 or B_2 do not exist, or $B_1 = B_2$, or $P_1 = 0$, or $P_2 = 0$.
- If after relocating to the *move* location, the set of “interesting” neighbors of the current beacon, as defined next, would not be a subset of the neighbors of the beacon at the new location. Notionally this can be done by moving the ant to the location, testing, then moving back.
- If the new *move* location is not reachable from the current location due to an obstacle or environmental border. Notionally this can be done by moving the ant to the location, testing, then moving back.

Else return *true*.

An “interesting” beacon is one which is likely to be part of an important path. We'd prefer to not damage such paths. At present we test for such beacons conservatively based on how often they've been used (their *wander* pheromone). We define two new constants: WANDERFRACTION (0.7) and MINWANDER (200). A beacon is “interesting” if its *wander* pheromone is $\leq \text{WANDERFRACTION} \times W$ (it's not been very much) and if the *wander* pheromone of the current beacon is $\leq \text{MINWANDER}$ (the region is old enough to have reliable wander statistics).

Move(current beacon). Move the beacon, and the ant, to the midpoint location computed by CanMove().

CanRemove(current beacon). There are two cases in which we presently remove beacons: first, if the beacon appears to be stranded or at the end of an abandoned string, and second, if the beacon is redundant.

For the first test, we return *true* if the neighborhood of the current beacon contains more than 2 other beacons, and the current beacon is not within the range of the food or nest, and it's old enough (its *wander* pheromone is $\leq \text{MINWANDER}$).

For the second test, we return *true* if there is another beacon (breaking ties randomly), called the *merge beacon*, within the range of the current beacon which has both higher *foraging* and *ferrying* pheromones, and which is within range of everything (food, nest, other beacons) that are within the range of the current beacon.

If we fail both tests, we return *false*.

Remove(current beacon). Remove the current beacon. Set the *wander* pheromone of the merge beacon (if any) to the minimum of the merge beacon and the current beacon.

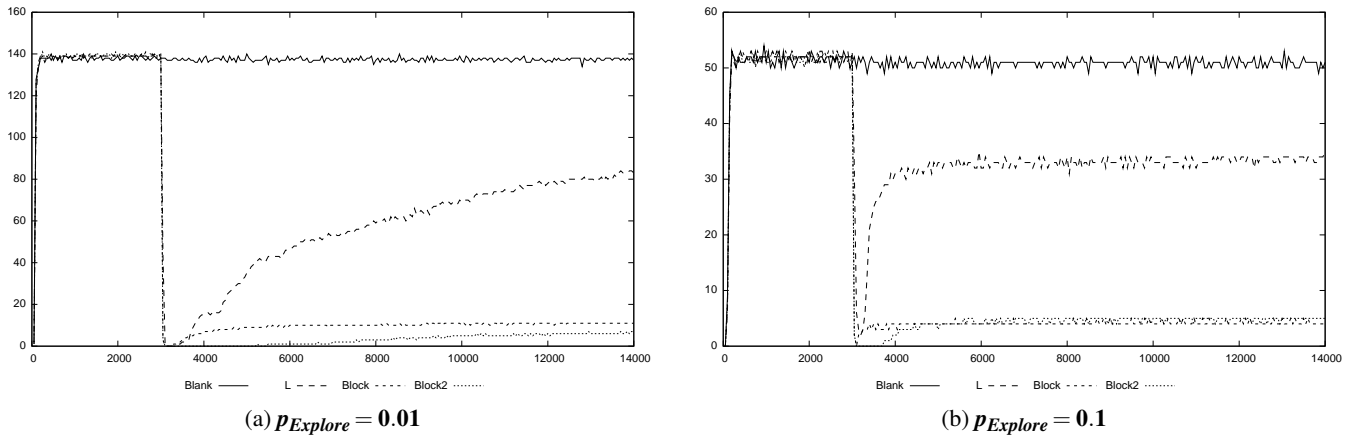


Figure 5: Mean food collection for the four static environments adding obstacles. $p_{Move} = 0$

3.4 Example

An example trace of the algorithm is shown in Figure 2. The ants optimize the trail in two basic ways. First, they may adopt a new route through the established beacons, or newly deployed beacons. Second, they can move beacons and eventually remove beacons which are “sufficiently close” to one another. Eventually the ants establish a reasonably optimized route between the food and nest, abandoning suboptimal routes and tightening up the best-discovered route.

Note that the path will likely never be fully optimized in this example because our present beacon-deployment and beacon-movement rules are overly conservative: the beacon-movement rule tries at all cost to avoid breaking chains; and the beacon-deployment rule winds up refusing to deploy in certain situations it perceives as overly crowded, even though they are needed to improve the current route.

4. EXPERIMENTS

We tested our algorithm to demonstrate the ant’s ability to find the food, to discover optimal transition sequences between the nest and the food, and to recover when obstacles are added to the environment. Our metric was the amount of food collected by all the ants every 50 timesteps.

Obstacles. To perform our experiments, we constructed several different obstacle environments, which are shown in Figure 3. We chose these environments to test two aspects of the algorithm: adaptively searching through the beacon graph (Exploration), and moving beacons so as to optimize the path when allocated only a limited budget of them (Optimization).

The *L* obstacle allowed ants to deploy beacons near the food source but forced them to create a path around the edge of the environment. The ants could effectively explore the majority of the landscape. The *Block* and *Block2* obstacles forced ants into narrow corridors, and occupied much of the landscape. Finally, the *Ant Clock With Obstacle* environment, discussed later, included a large obstacle to complicate a dynamic environment. We compared each of these obstacles with a *Blank* environment as a control. For *L*, *Block*, *Block2*, and *Blank* the nest was placed in the upper left corner at (10, 10), and the food in the lower right corner at (90, 90).

Minutiae. We ran 63 independent runs in the MASON simulator [9] for 14,000 timesteps each. p_{Follow} and p_{Deploy} were fixed

at 0.9. All experiments used 100 ants, a 100x100 bounded, continuous world, and a beacon range of 10. We limited the number of available beacons to 60 for *Block*, 100 for *Block2*, and 400 for all other environments. These limits provide just enough beacons to establish a trail for the given environment. We compared results after 2,000 and 14,000 timesteps using an ANOVA test with a Tukey post hoc comparison at the 95% confidence level.

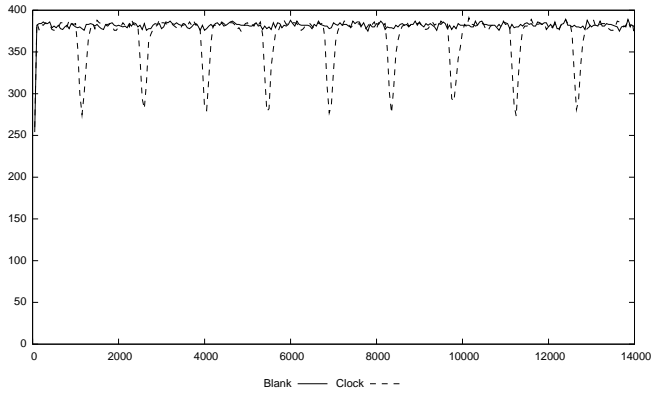
4.1 Exploration

We began by examining the ants’ ability to find better routes through the graph given beacons which, once deployed, could not be moved or deleted. Accordingly, we set $p_{Move} = 0$, and $p_{Explore}$ varied over 0.0001, 0.001, 0.01, 0.1, 0.5, and 0.9. We studied two cases: removing obstacles (which created new situations to exploit) and adding new obstacles (which had to be worked around).

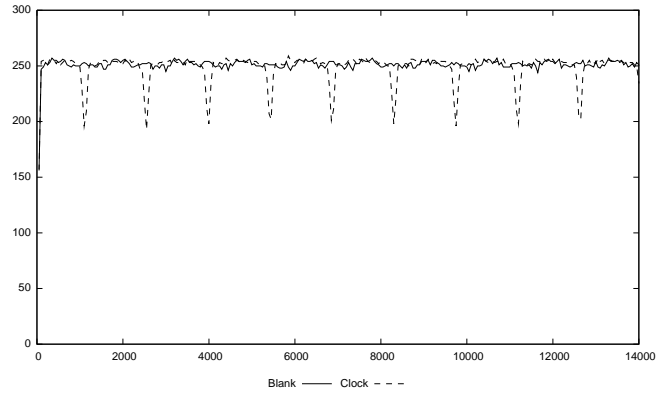
Removing obstacles. We began by letting the ants discover a suboptimal trail around some obstacle for 3,000 timesteps, then removed the obstacle to see how the ants would find new routes to take advantage of their revised situation. Our obstacle of choice for this experiment was the *L* obstacle. Figure 4 shows the performance with $p_{Explore} = 0.1$ before and after removing the *L* obstacle. As can be seen, the ants rapidly adapt to the new situation. After the obstacle is removed, performance converges rapidly to approximately the same performance of the ants on the *Blank* environment, as the ants find a superior path through the beacon graph. Changing $p_{Explore}$ does not significantly alter this rate of adaptation: though larger values of $p_{Explore}$ generally result in significantly lower total food collection as more time is spent exploring.

Adding obstacles. Next we examined how the ants would react to environmental changes that made previously good trails no longer viable. We let the ants explore an empty environment for 3000 time steps, and then introduced obstacles. When an obstacle was introduced, it might collide with a number of beacons and ants. We treated these as destructive events to the beacons and ants. Specifically, a beacon in collision was automatically removed. An ant in collision was “killed” — it was eliminated entirely, and so the total count of ants was reduced by one. We chose to do this rather than artificially “restart” the ant at the nest or “move” it to a safe location.

Figure 5 shows the ants’ performance for two values of $p_{Explore}$. In both cases we see that the ants can recover, but their performance after the obstacle is introduced is reduced proportionally to



(a) $p_{Explore} = 0.01$



(b) $p_{Explore} = 0.1$

Figure 6: Effect of encountering and adapting to an obstacle while following a moving food source. $p_{Move} = 0$

the number of resources (ants and beacons) that were destroyed by the obstacle. The L obstacle, which exhibits the best recovery, covers a smaller area than either Block or Block2, both of which show more limited performance.

In comparing the two graphs, it is important to note the scale. While the ants recover faster with a higher $p_{Explore}$ value, their overall performance is less than the ants with a lower $p_{Explore}$. This is essentially the same situation as noted in the previous experiment (removing obstacles): as the ants spend more time exploring (represented by higher $p_{Explore}$ values), they spend less time ferrying food to the nest. As $p_{Explore}$ approaches 1, the performance of the ants drops dramatically as they spend more time exploring.

4.2 Dynamic Food Location

Having shown that the ants could adopt better trails where the food location was not moving, we next tested to see if this held when the food was moving. To do this, we recreated an experiment performed in [16], called the ‘‘Ant Clock’’. In this experiment, the nest started in the center of the environment (50, 50), and the food was initially placed due east, 10 distance units from the right edge of the environment. At each timestep, the food would rotate about the nest in a clockwise direction at one-quarter of a degree per timestep. We placed an obstacle north of the nest such that the food would just clear the left and right edges of the obstacle in its orbit about the nest (see Figure 3). As a control we had the food source rotate about the nest, but without the obstacle to the north. We set $p_{Move} = 0$, and varied $p_{Explore}$.

Without the obstacle, the ants ably adapted to the constantly moving obstacle, maintaining an approximately straight-line path at all times. With the obstacle, the ants’ path would effectively ‘‘bend’’ around the obstacle as the food passed by it, but eventually the exploration would enable the ants to reestablish an optimized straight path. This ‘‘bending’’ is reflected in the periodic drops in performance in Figure 6. Figures 6(a) and 6(b) illustrate another tradeoff of more exploration: higher values of $p_{Explore}$ decrease the absolute amount of food returned the nest; but higher $p_{Explore}$ values also decrease the severity of the periodic drops. The increased exploration prevents the ants from spending too much time on the established, suboptimal ‘‘bent’’ trail.

4.3 Optimization

In our final experiment, we set $p_{Explore} = 0$ and varied p_{Move} test the ants’ ability to optimize trails with a limited number of beacons. Here we used the Block and Block2 environments. We

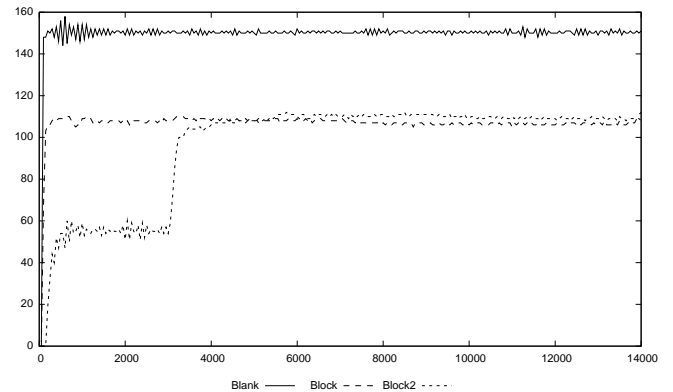


Figure 7: Mean food collected for the Block and Block2 environments with $p_{Explore} = 0, p_{Move} = 0.1$

sought to test one of the key ideas behind our algorithm: that a sparse representation of pheromones benefits from physical revisions and updates. In the experiment, let the ants establish a trail for 3,000 timesteps and then removed the obstacle.

After removing the obstacle, the ants would begin to move the beacons so as to straighten the trail, and eventually straighten the trail entirely, deleting redundant beacons as the trail became shorter. For $p_{Move} = 0.1$, Figure 7 shows that while performance statistically improves after the obstacle is removed, it does not converge to the performance of Blank. We can surmise two possible reasons for this: first, though beacons would be removed, the trail was ultimately still denser with beacons than if the ants had been (as in Blank) free to deploy beacons in the space. Second, moving and deleting beacons would occasionally trap ants in ‘‘islands’’ — small disjoint beacon groups — and unable to participate.

Even so, the results on the Block2 environment verified our visual inspection that the trail line was rapidly straightened out and optimized. In the Block environment this effect is not seen, largely because the number of beacons remained approximately the same before and after optimization. Similar performance is seen with $p_{Move} = 0.5$ and $p_{Move} = 0.9$.

5. CONCLUSION

We presented an approach to establishing trails among swarm ant robots using a non-invasive and non-destructive stigmergic communication in the form of deployable and movable beacons. The

robots use those beacons as a sparse representation of a pheromone map embedded in the environment. The algorithm uses a variation of value iteration to update pheromones and make transitions from beacon to beacon. We demonstrated the efficacy of the technique and explored its present robustness, and optimization capabilities.

This work is intended as a stepping stone to actual deployment on swarm robots, and using sensor motes as beacons. This deployment is our first task in future work. We will also examine extending the beacon model to more collaborative tasks than simply establishing trails. For example, in other experiments we have demonstrated sophisticated self-crossing, multi-waypoint tours. We believe we can also employ beacons in this model to define regions to avoid or requests for assistance (to move objects or establish formations, for example).

6. REFERENCES

- [1] E. Barth. A dynamic programming approach to robotic swarm navigation using relay markers. In *Proceedings of the 2003 American Control Conference*, volume 6, pages 5264–5269, June 2003.
- [2] E. Bonabeau. Marginally stable swarms are flexible and efficient. *Phys. I France*, pages 309–320, 1996.
- [3] E. Bonabeau and F. Cogne. Oscillation-enhanced adaptability in the vicinity of a bifurcation: the example of foraging in ants. In P. Maes, M. Matarić, J. Meyer, J. Pollack, and S. Wilson, editors, *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior: From Animals to Animats 4*, pages 537–544. MIT Press, 1996.
- [4] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 1999.
- [5] C. Chibaya and S. Bangay. A probabilistic movement model for shortest path formation in virtual ant-like agents. In *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, pages 9–18, New York, NY, USA, 2007. ACM.
- [6] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Insect Behavior*, 3:159–168, 1990.
- [7] P. P. Grassé. La reconstruction du nid et les coordinations inter-individuelles chez *Bellicositermes natalensis* et *Cubitermes* sp. La theorie de la stigmergie: Essai d'interpretation des termites constructeurs. *Insectes Sociaux*, 6:41–80, 1959.
- [8] L. R. Leerink, S. R. Schultz, and M. A. Jabri. A reinforcement learning exploration strategy based on ant foraging mechanisms. In *Proceedings of the Sixth Australian Conference on Neural Networks*, Sydney, Australia, 1995.
- [9] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. MASON: A multiagent simulation environment. *Simulation*, 81(7):517–527, July 2005.
- [10] N. Monekosso, P. Remagnino, and A. Szarowicz. An improved Q-learning algorithm using synthetic pheromones. In E. N. B. Dunin-Keplicz, editor, *Second International Workshop of Central and Eastern Europe on Multi-Agent Systems*, Lecture Notes in Artificial Intelligence LNAI-2296. Springer-Verlag, 2002.
- [11] N. D. Monekosso and P. Remagnino. Phe-Q: a pheromone based Q-learning. In *Australian Joint Conference on Artificial Intelligence*, pages 345–355, 2001.
- [12] M. Nakamura and K. Kurumatani. Formation mechanism of pheromone pattern and control of foraging behavior in an ant colony model. In C. G. Langton and K. Shimohara, editors, *Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, pages 67–76. MIT Press, 1997.
- [13] K. O'Hara and T. Balch. Distributed path planning for robots in dynamic environments using a pervasive embedded network. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1538–1539, 2004.
- [14] K. O'Hara, V. Bigio, S. Whitt, D. Walker, and T. Balch. Evaluation of a large scale pervasive embedded network for robot path planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2072–2077, May 2006.
- [15] K. O'Hara, D. Walker, and T. Balch. *The GNATs—Low-cost Embedded Networks for Supporting Mobile Robots*, pages 277–282. Springer, 2005.
- [16] L. Panait and S. Luke. A pheromone-based utility model for collaborative foraging. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 36–43, Washington, DC, USA, 2004. IEEE Computer Society.
- [17] H. Parunak, S. Brueckner, and J. Sauter. Synthetic pheromone mechanisms for coordination of unmanned vehicles. In *Proceedings of First Joint Conference on Autonomous Agents and Multi-agent Systems*, pages 449–450, 2002.
- [18] H. V. D. Parunak, S. A. Brueckner, and J. Sauter. *Digital Pheromones for Coordination of Unmanned Vehicles*, pages 246–263. Springer Berlin, 2005.
- [19] D. Payton, M. Daily, R. Estkowski, M. Howard, and C. Lee. Pheromone Robotics. *Autonomous Robots*, 11(3):319–324, 2001.
- [20] M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, 1994.
- [21] R. Russell. Laying and sensing odor markings as a strategy for assisting mobile robot navigation tasks. *Robotics and Automation Magazine*, 2(3):3–9, Sep 1995.
- [22] R. T. Vaughan, K. Støy, G. S. Sukhatme, and M. J. Matarić. Blazing a trail: insect-inspired resource transportation by a robot team. In *Proceedings of the International Symposium on Distributed Autonomous Robot Systems*, 2000.
- [23] R. T. Vaughan, K. Støy, G. S. Sukhatme, and M. J. Matarić. Whistling in the dark: Cooperative trail following in uncertain localization space. In C. Sierra, M. Gini, and J. S. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 187–194. ACM Press, 2000.
- [24] R. T. Vaughan, K. Støy, G. S. Sukhatme, and M. J. Matarić. LOST: Localization-space trails for robot teams. *IEEE Transactions on Robotics and Automation*, 18:796–812, 2002.
- [25] M. Wodrich and G. Bilchev. Cooperative distributed search: The ants' way. *Control and Cybernetics*, 26, 1997.
- [26] V. A. Ziparo, A. K. B. Nebel, and D. Nardi. RFID-based exploration for large robot teams. In *Proceedings of International Conference on Robotics and Automation*, pages 4606–4613, 2007.