Ant Colony Algorithm for Swarm Systems

Víctor Soto Hernández and Alfredo Weitzenfeld, *Senior Member, IEEE*

***Abstract*—The paper presents a swarm model inspired in ant colonies where ants perform different tasks. The model emphasizes explorer and worker ants responsible for food search and bringing food back to the nest, respectively. The model is based on ACO – Ant Colony Optimization providing a shortest path algorithm during route exploration. Results from ant colony model simulations are presented and contrasted.**

***Keywords*—Ant colony, multi-agent systems, behavior modeling.**

I. Introduction

ANT COLONY OPTIMIZATION (ACO) is an intelligent colony algorithm inspired by swarm behavior in real ants [1]. Extensive work has been done based on swarm studies producing a number of models and corresponding applications intended to solve optimization problems such as routing in telecommunications and the traveling salesman or postman problem in which an optimal route must be calculated to deliver packets or letters in different locations.

Swarms define groups or teams of agents that share a common goal. Team members coordinate behaviors by adapting cognitive processes to their own perceptions (input sensing) and communications, direct or indirect, with other team members [2].

The ant is the insect that best represents the concept of *swarm intelligence* providing inspirations to distributed problem solving. Originally, swarm studies were inspired by biological social behaviors in insects such as termites, wasps or ants. Skills shown by swarms seem to exceed skills shown by individual agents. Thus intelligent swarms display intelligent group *emergent behavior* even if individual members are not considered themselves intelligent. In many such biological studies, high level group behavior is achieved by a small set of simple low level interactions among individuals and their environment.

Among the many properties that can be found in an *Intelligent Colony* it is important to emphasize:

- **Distribution**. A colony is distributed, i.e., there does not exist a central control entity or a central source of information.
- **Parallelism**. Colony members may process in parallel improving overall task performance. If tasks can be further subdivided benefits are even larger.
- **Robustness**. Ants or agents in the colony may fail and even die without affecting the overall task. This is due to redundancy where agents can adapt to changes in the environment including varying number of agents.
- **Modularity**. The problem can be analyzed by parts such as by specific behaviors performed by colony members.
- **Scalability**. Solutions to simpler problems can be extended to larger ones.
- **Autonomy**. Colony individuals can sense and modify environments.

II. Ant Colony Optimization

An ant colony consists of a complex social structure in which each ant type has its own functions to do in the rest if its life [3]. Ants in the colony include a hierarchical organization:

- *Queen* ant responsible for laying eggs for future colony members.
- *Soldier* ants take care of the whole colony defending the nest from other insects, including ants belonging to other colonies.
- *Explorer* ants look for food leaving pheromone traces in their path.
- *Worker* ants follow pheromone path left by explorers ants.

There exist many ant-based algorithms, nevertheless the most important one is ACO algorithm. ACO is inspired by the experiment described in Figure 1. An ant colony has access to a food source through a bridge which has two paths, one longer than the other. After certain period of time, most of the ants will end up going through the shortest path. Furthermore, the probability of choosing this shortest path increases in proportion to the difference between the shorter and longer path distances. Ants have the ability to modify the environment they explore by leaving tracks of *pheromone* on the soil. In a decision point (path bifurcation) the probability of selecting one particular path is based in the quantity of pheromone they sense on each path. Ants choose the shortest path by following paths marked by other paths that have got to the food source and back to the nest in the least amount of time. Followings ants sense more pheromone in the path increasing the probability for the next ant choosing the same shortest path.

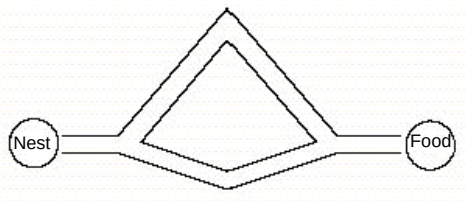

Fig. 1. Bridge between nest and food source.

1-4244-0537-8/06/$20.00 ©2006 IEEE

In the ACO algorithm the main task is for each ant to find the shortest path between a source node *s* and a destination node *d* as highlighted in blue and shown in Figure 2. In these problems route length is computed by minimizing the number of hops the ant has to take.

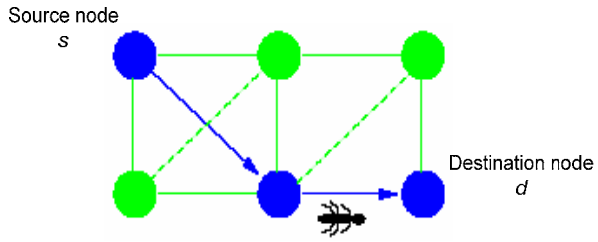

Fig. 2. Optimal route of length two from source node *s* to destination node *d*.

For each arc *i,j* that interconnects nodes *i* and *j* in the graph we define a variable $\tau_{i,j}$ representing the *artificial pheromone* or *pheromone trail*. This trail is written and read by each ant. The quantity of pheromone that ants leave in the explored surface is proportional to the utility of each arc estimated by each ant. The probability of ant *k* at node *i* of moving to node *j* is represented by $k_{i,j}$. Equation 1 describes the probability calculation, where node *j* belongs to the set *N* containing all one-step neighbors of node *i*.

$$p_{i,j}^{k} = \begin{cases} \dfrac{\tau_{i,j}}{\sum_{a \in N_i^k} \tau_{i,a}} & if \;\; j \in N_i^k \\ 0 & if \;\; j \notin N_i^k \end{cases} \qquad (1)$$

If we consider an ant located at node *i* with five neighboring nodes *f, j, k, l, m,* and *n*, then an arc that connects node *i* with the new nodes an artificial pheromone value τ as shown in Figure 4.

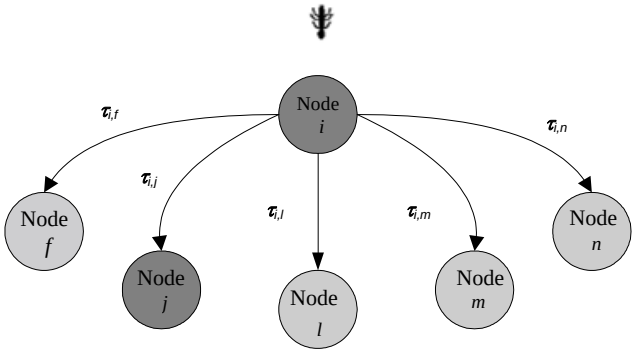

Fig. 3. Artificial pheromone values for arcs connecting node *i* with five neighboring nodes *f, j, k, l, m,* and *n*.

The five neighboring nodes *f, j, k, l, m,* and *n*, correspond to the five neighboring cells in a two dimensional grid as shown in Figure 5. Note that only the front and side cells are considered as next moving steps.

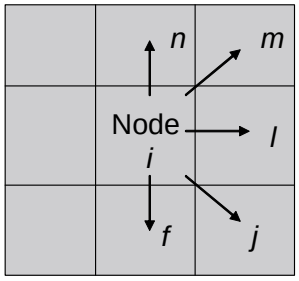

Fig. 4. Neighboring cells *f, j, k, l, m,* and *n* for node *i*.

The probability of ant *k* moving from node *i* to node *j* in Figures 3 and 4 is described by equation 2.

$$p_{i,j}^{k} = \frac{\tau_{i,j}}{\sum_{a \in N_i} \tau_{i,a}} \qquad (2)$$

In this algorithm ants update traces by $\Delta\tau$ pheromone after each step. After a period of time *t*+1, the amount of pheromone at a particular point is described by equation 3,

$$\tau_{i,j}(t+1) = \tau_{i,j}(t) + \Delta\tau \qquad (3)$$

According to this rule, an ant using arc *i,j* will increase the probability of subsequent ants of choosing the same path. The algorithm requires selecting an appropriate value for $\Delta\tau$. There are two ways of updating the pheromone trail, a *constant trace* and *variable trace*.

Constant trace. In the simplest case and similarly to real ants, pheromone update $\Delta\tau$ is kept constant all the time, i.e. same values are used for every arc the ant travels. The shortest path is computed by *differential path length*, i.e. shorter paths are computed by stronger pheromone concentrations. This occurs since ants will strengthen shorter paths by leaving pheromone traces more often than in longer paths. Thus, the probability of ants choosing the shortest path is incrementally increased.

Variable trace. A more complex way of updating pheromone trail sets $\Delta\tau$ dependent on path length. A pheromone trace will be computed by a variable trace function $\Delta\tau = 1/L_k$, where L_k is the length of the path found by ant *k*. If a path is shorter then the amount of pheromone trace will be higher. To avoid a quick convergence a pheromone evaporation mechanism is included. Thus, the intensity of pheromone decreases improving the probability of exploring new routes. In general, pheromone evaporation is important in artificial ants since they offer a learning mechanism for newer policies to forget bad decisions made in the past.

III. Behavior Model

In order to simulate an ant colony we established a set of basic behavior objectives and rules:

- The ant colony goal is to find a route from the nest to a food source and vice versa, while storing the food in

the nest as quick as possible.

- The ant colony will be conformed by two types of ants: explorers and workers.
- Explorer ants search for a path from nest to food source. Food gathered is taken back to the nest using the same path.
- Worker ants follow one of the paths discovered by explorer ants.
- After some time most of the workers should follow a single unique path from nest to food source. It must be one of the shortest paths found by the explorers.
- Communication among ants is limited to pheromone traces indirect laid on the exploring surface.

In the next two sections we describe behaviors for explorer and worker ants. There are some similarities between the two.

A. Explorer Ants

The *Explorer* ant life cycle is described as follows:

1. The explorer ant gets out of the nest walking in an arbitrary path in search of food.
2. During the search the explorer ant modifies the environment by laying pheromone traces on the soil in order to attract other explorers to its route.
3. When food is discovered the explorer ant takes some and returns to the nest following the same path it used to get there.
4. Each explorer can repeat the search twice to avoid saturation of ants in the exploring surface.

The explorer behavior states are shown in Figure 5 and described in more detail as follows:

Exploring. Before the explorer leaves the nest, it senses the quantity of pheromone in each node (cell) around. The next cell is chosen by means of the probabilistic rule previously explained in the ACO algorithm. When the ant moves to the next cell, it calculates again the probability and moves to the following cell and so on.

Coming back to nest. When a food source is discovered, the explorer takes a portion of food and returns to the nest using the same path back. The state ends when the nest is perceived.

Using path. The explorer ant searches for food again similarly to *Exploring* state. The state ends when the nest is perceived.

Returning. The explorer ant returns to the nest using the same previous path. This state is very similar to *Coming back to nest* state. The difference is that the *Returning* state ends when either the explorer finds the nest and goes again to the food source, so next state would be *Using path*, or the explorer has already used the path *n* times terminating in *Death* state.

Returning without food. If the ant has explored for an extensive period of time without finding any food, it returns to the nest using the learnt path and terminating in *Death* state.

Death. In this state explorers stop their search, i.e. explorations ends.

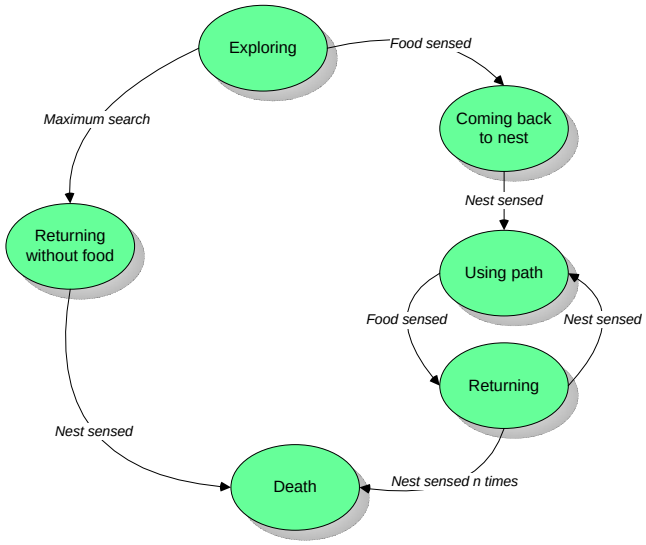

Fig. 5. Explorer ant behavior described by a state machine comprising *Exploring*, *Coming back to nest*, *Using path*, *Returning*, *Returning without food* and *Death*.

B. Worker Ants

The *Worker* ant life cycle is described as follows:

1. The worker ant gets out the nest to bring food back to the nest.
2. It must follow one of the routes discovered by the explorers.
3. When a worker has got to the food nest it brings the food back to the nest using the same path.
4. Workers choose the shortest routes from the different routes discovered by the explorers.
5. Workers also affect the environment so other workers can follow the same shortest route.

The worker behavior states are shown in Figure 6 and described in more detail as follows:

Birth. A worker ant chooses from one of the routes explorers discovered. The decision is based on the probabilistic rule defined by the ACO algorithm. Each route is assigned a probability based on the corresponding pheromone value. For example if route *A* has a greater pheromone value than route *B*, then route *A* will have a grater probability of being selected by worker ants.

Using route. Worker follows to the route to the food source previously chosen. This sate ends when the food is sensed.

Returning. This state is similar to *Returning* state for explorers ants. The main difference is that a worker ant can travel through different paths. The state ends when the worker arrives to the nest.

New route. Each worker ant can choose from any route discovered by the explorers. Workers also update pheromone traces for the traveled route.

Death. In this state workers stop bringing back food, i.e. work ends.

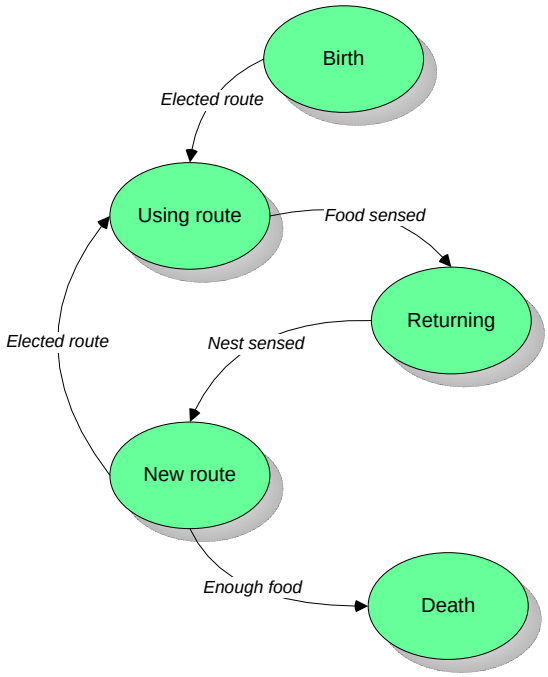

Fig. 6. Worker ant behavior described by a state machine comprising *Birth*, *Using route*, *Returning*, *New route* and *Death*.

IV. Simulation System

The ACO algorithm and behavior model described in sections II and III were simulated using the *Swarm* system [4]. *Swarm* is a software platform for adaptive multi-agent system simulation with special emphasis on ant colonies type of models. The main objective of this system is to model and simulate large numbers of computational objects or agents implementing simple behaviors. The simulation system involves specifying and implementing a *Behavior* and *Graphic Display* modules as described in the following sections.

A. Behavior Module

In the *Behavior Module* we specified a special object class called *Ant* where we defined all behavior states. The *Ant* class includes attributes corresponding to ant localization in the simulated world. A direct mapping between behaviors and functions was done to implement each state. The model includes a *PheromoneSpace* class managing pheromone value updates. Additionally, an *AntModelControl* class manages the full simulation model. The *Ant* and *AntModelControl* class interfaces in the *Behavior Module* are shown in Figure 7.

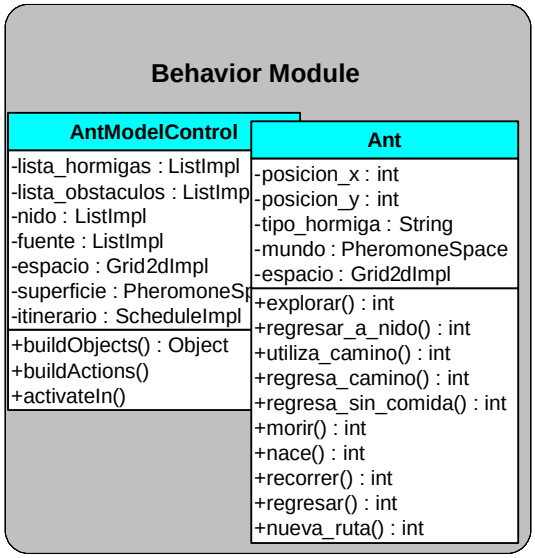

Fig. 7. The diagram shows *Ant* and *AntModelControl* class interfaces in the *Behavior Module*.

B. Graphic Display Module

The *Graphic Display Module* allows an easy control of graphic actions independent from behaviors. The module includes an *AntObserverControl* specifying nest, food source and obstacles in the environment as shown in Figure 8.

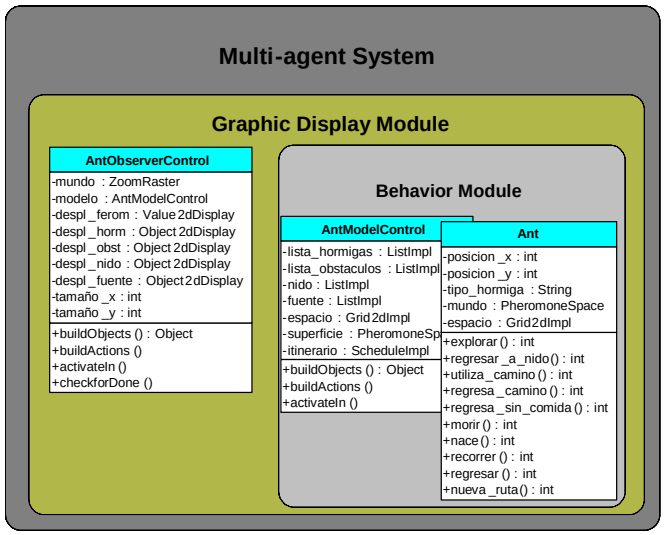

Fig. 8. The diagram shows *AntObserverControl* class interfaces in the *Graphic Display Module*.

Figure 9 shows the simulation system interface developed for this particular application.

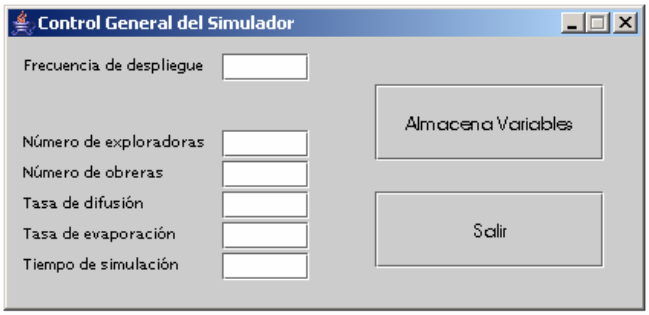

Fig. 9. Simulation system interface developed for this particular application.

V. Experiments and Results

In order to make a detailed analysis of the behavior model we specified a number of experiments and defined a set of parameters to modify for each experiment.

Figure 10 shows an environment consisting of a nest in blue, a single food source in yellow, and a number of obstacles in grey with explorer ants highlighted in red. In the figure, explorer ants are just leaving the nest.

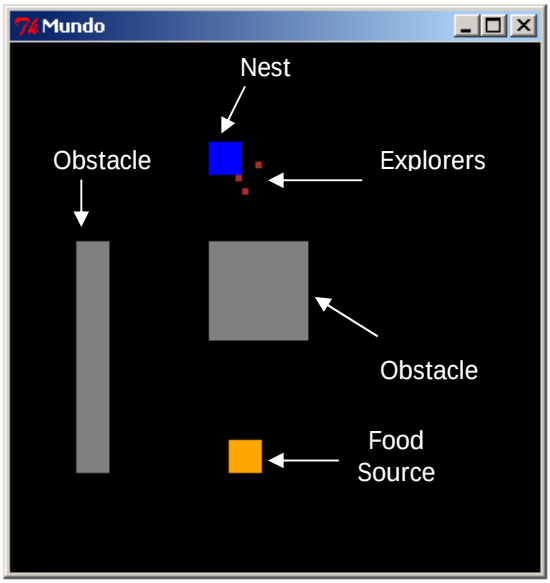

Fig. 10. Virtual world consisting of a nest in blue, a single food source in yellow, and a number of obstacles in grey with explorer ants highlighted in red. Explorer ants are just leaving the nest.

Figure 11 shows an environment consisting of a nest in blue, three food sources in yellow, and a number of obstacles in grey with explorer ants highlighted in red. Source number 2 is closest to nest while sources 1 and 3 are further away.

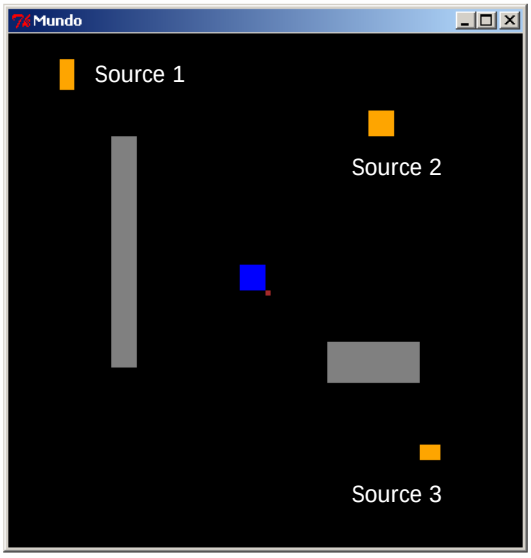

Fig. 11. Virtual world consisting of a nest in blue, three food sources in yellow, and a number of obstacles in grey with explorer ants highlighted in red. Source number 2 is closest to nest while sources 1 and 3 are further away.

Figure 12 shows an environment consisting of a nest in blue, a food sources in yellow, and a number of obstacles in grey. Explorer ants highlighted in red correspond to ants in state *exploring* (or *using path*) with explorer ants in yellow representing ants in *coming back to nest* (or *returning*) states.

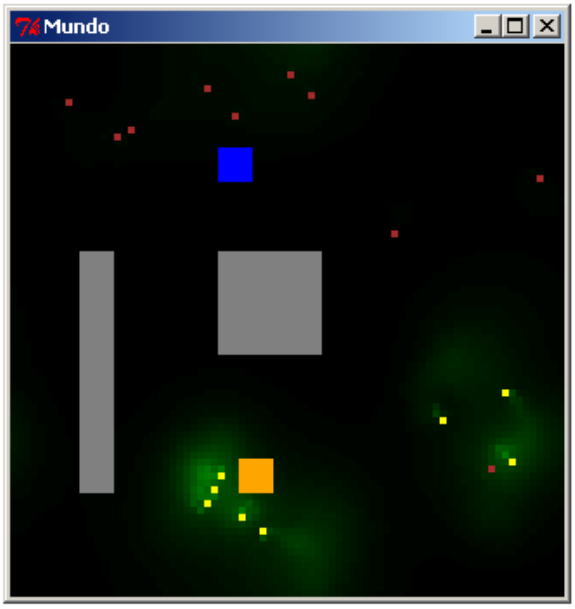

Fig. 12. Virtual world consisting of a nest in blue, a single food source in yellow, and a number of obstacles in grey. Explorer ants highlighted in red correspond to ants in state *exploring* (or *using path*) with explorer ants in yellow representing ants in *coming back to nest* (or *returning*) states.

We defined a set of four simulations to help analyze explorer and worker behaviors. Simulations vary with respect to a number of parameters as described in Table I.

TABLE I
SIMULATION SETS

Set	Explorer Ants	Evaporation Rate
A	30	0.5
B	30	0.001
C	75	0.5
D	75	0.001

When evaporation rate is large pheromone traces evaporate faster. In each set of simulations we experiment with different sets of diffusion rates. If the diffusion rate is large the pheromone will spread out more quickly in the exploration surface. For each set of diffusion rates we ran 50 simulations to find a shortest route. When the 50 simulations are finished we get an average of the shortest route length. We analyze in the explorer behavior if the resulting shortest route reaches the actual closest food source. Figure 13 shows results for the four sets of simulations.

As seen in the graph, the best results are obtained with an increased number of explorer ants, corresponding to sets C and D having shorter average routes. By having more explorers, the number of routes to food sources increases, resulting in a higher probability of finding shorter routes than with smaller number of explorers.

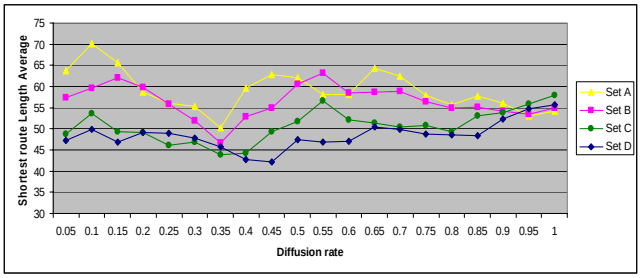

Fig. 13. The diagram shows average shortest routes obtained for simulation sets defined in Table I. Diffusion rates are varied during the experiments.

Nevertheless, quality of exploration can improve even when the number of explorers is kept constant if pheromone traces remain in the surface more time. We achieve this by increasing diffusion rates. The worst results occur for diffusion rate values between 0.05 and 0.15 with the faster evaporation rate of 0.5 corresponding to sets A and C compared to sets B and D, respectively. The worst overall case is set A, having limited pheromone concentrations and few explorers.

Workers need to converge to shortest route paths found by explorers. To make all workers follow the same route they have to update pheromone value for the corresponding route. We have developed to groups of simulations corresponding to either constant or variable trace updates. We compare in Table II the percentage of simulations that converge to (1) the shortest route, (2) the second and third shortest routes, and (3) forth and other shortest route. We ran one hundred simulations for each type of trace update using 75 explorers and an evaporation rate of 0.001 corresponding to set D having the best parameter combination in Figure 13.

TABLE II
SHORTEST ROUTE CONVERGENCE WITH 75 EXPLORERS

	Shortest Routes	Constant Traces	Variable Traces
1	First	69 %	89 %
2	Second y Third	17 %	9 %
3	Forth and other	14 %	2 %

To increase the percentage of shortest route convergence for a constant update we ran 100 additional simulations with 150 explorers as shown in Table III.

TABLE III
SHORTEST ROUTE CONVERGENCE WITH 150 EXPLORERS

	Shortest routes	Constant Update
1	First	84 %
2	Second and Third	12 %
3	Forth and other	4 %

The improvement can be explained by the fact that increasing the number of explorers increases the total number of routes to reach food sources.

VI. Conclusion

The paper presented a swarm model based on ant colonies behavior. The model concentrates on exploration of food sources and carrying food back to the nest. The paper discusses the ACO – ant colony optimization algorithm used to compute probabilities from constant and variable pheromone trace approaches. Behaviors for explorer and worker ants are described. Results are presented comparing simulations based on different numbers of explorer ants, different number of pheromone evaporation and diffusion rates.

While some interesting results were obtained, the model can be further extended with more in-depth parameter analysis. There are many limitations in the model such as the restricted number of cells that an ant can move during each step, the limited set of parameter ranges having been evaluated, the possibility of collisions between explorers, and the limited analysis of the probability-based algorithms.

Current work concentrates on implementing the swarm models in real robots to experiment under real constraints, such as limited sensory information, among others.

Acknowledgment

This work has been supported in part by UC-MEXUS CONACYT, CONACYT grant #42440, LAFMI, and “Asociación Mexicana de Cultura A.C.” in Mexico.

References

[1] Dorigo, Marco and Stützle, Thomas. *Ant Colony Optimization*. The MIT Press, 2003.
[2] Stone, Peter. *Layered Learning in Multi-agent Systems: A Winning Approach to Robotic Soccer*. The MIT Press 2000.
[3] Holldobler, Bert and Wilson Edward O., *The Ants*, Belknap Press, 1990.
[4] Swarm Release 2.2 Documentation. http.//wiki.swarm.org