Learning Ant Foraging Behaviors

Liviu A. Panait and Sean Luke
George Mason University, Fairfax, VA 22030
lpanait@cs.gmu.edu, sean@cs.gmu.edu

Abstract

Insects are good at cooperatively solving many complex tasks. For example, foraging for food far away from a nest can be solved through relatively simple behaviors in combination with pheromones. As task complexity increases, however, it may become difficult to find individual agent rules which yield a desired emergent cooperative behavior, or to know if any such rules exist at all. For such tasks, machine learning techniques like evolutionary computation (EC) may prove a valuable approach to searching the space of possible rule combinations. This paper presents an application of genetic programming to search for foraging behaviors. The learned foraging behaviors use only pheromone information to find the path to the nest and to the food source.

Introduction

Artificial Intelligence has drawn many ideas from biology: evolutionary computation, neural networks, robotics, vision, and cooperative problem solving all steal liberally from Mother Nature. One such area of particular recent interest in AI has been in algorithms inspired from social insects such as ants, termites and bees. The interest stems from the capacity of such simple organisms to collaboratively work together to solve problems no one individual could. Some social-insect-inspired AI literature has focused on foraging and related tasks through the use of pheromones (Bonabeau et al., 1999). The social memory mechanism of pheromones is an inviting paradigm for designing multiagent systems with blackboards, joint utility tables, and other global memory mechanisms. However, hand-coding of agent behaviors using this paradigm can prove problematic given the unexpected emergent group behaviors that arise.

While previous work has applied machine learning methods to the *use* of pheromone information, they have still tended to hard-code the pheromone depositing procedure. In contrast, this paper shows that it is possible to have the entire foraging behavior discovered by the learning system.

The paper proceeds with a description of previous work in learning foraging behaviors, and a description of an evolutionary computation approach to the learning task. A set of three experiments in increasingly difficult environments shows that good foraging behaviors can be discovered. A later experiment shows that behaviors learned for complex domains are robust to simpler environments.

Previous Work

The specific problem at hand is called *central place food foraging*, and it consists of two main phases: an initial exploration for food, followed by carrying it back to the nest (Sudd and Franks, 1987). When an ant reaches a food source, the ant automatically becomes laden with food; and when the ant reaches the nest, it automatically drops the food off at the nest.

Various learning algorithms have been used to attack this problem. Some algorithms related to reinforcement learning adopt a fixed pheromone laying procedure, then use the sensed pheromone information to explore the space or to update state-action utility estimates (Leerink et al., 1995; Monekosso et al., 2002). Evolutionary computation techniques have also been applied to learn exploration/exploitation strategies using pheromones deposited by hardcoded mechanisms. For example, Sauter et al show how EC can be used to tune the action-selection behavior in an application involving multiple "digital" pheromones (Sauter et al., 2002). A similar idea applied to network routing is presented in (White et al., 1998).

AntFarm (Collins and Jefferson, 1992) is another system that combines communication via pheromones and evolutionary computation, and it is the closest work to the algorithm presented in this paper. AntFarm uses multiple colonies of homogeneous ants, with each colony in a separate 16x16 grid environment. The ants use a single pheromone to mark trails to food sources, but use a compass to point themselves along the shortest path back to the nest. The system uses evolutionary computation to search for foraging behaviors represented as neural networks.

A trend common to all previously described algorithms is that the ants *know* how to return to the nest. This assumption is mainly based on observations that ants use sophisticated navigational techniques for this task, including orientation based on landmark memorization or using the position of

Pheromone Depositing Tree Function	Description
scalar ←CurFoodPhLevel()	Food pheromone at my location
scalar ←CurHomePhLevel()	Home pheromone at my location
scalar ←LastDeposited()	How much pheromone I deposited last time
scalar ←DistanceFromSite()	Number of time steps elapsed since I last visited the nest (or food, depending on state)
scalar ←MaxDistanceFromSite()	Max possible distance from site (depends on the maximum lifetime of ants)
scalar ←MaxLocalFoodPheromone()	Max food pheromone at my eight neighboring locations
scalar ←MinLocalFoodPheromone()	Min food pheromone at my eight neighboring locations
scalar ←MaxLocalHomePheromone()	Max home pheromone at my eight neighboring locations
scalar ←MinLocalHomePheromone()	Min home pheromone at my eight neighboring locations
scalar ←MaxPheromone()	Max amount of pheromone possible
scalar ←MaxPhomoneDividedByMaxDistanceFromSite()	MaxPheromone() / DistanceFromSite()
scalar ←MaxPheromoneDividedByMaxDistanceFromSite()	MaxPheromone() / MaxDistanceFromSite()
scalar ←Add(*scalar, scalar*)	Add two scalars
scalar ←Sub(*scalar, scalar*)	Subtract two scalars
scalar ←Max(*scalar, scalar*)	Maximum of two scalars
scalar ←Min(*scalar, scalar*)	Minimum of two scalars

Behavior Selection Tree Function	Description
vector ←FoodPheromones()	Amounts of food pheromones at the eight neighboring locations
vector ←HomePheromones()	Amounts of home pheromones at the eight neighboring locations
vector ←AddV(*vector, vector*)	Add two vectors
vector ←SubV(*vector, vector*)	Subtract two vectors
vector ←Mul2V(*vector*)	Multiply each component of a vector by 2
vector ←Div2V(*vector*)	Divide each component of a vector by 2
vector ←SqrV(*vector*)	Square each component of a vector
vector ←Sqrt(*vector*)	Take the square root of each component of a vector
direction ←MinO(*vector*)	Return the index of the smallest component of a vector
direction ←MaxO(*vector*)	Return the index of the largest component of a vector
direction ←ProbO(*vector*)	Return a random index, chosen using the normalized component sizes as probabilities (+ .001)

Table 1: Function set for an ant's GP pheromone-depositing and behavior-selection trees. Functions depicted take the form of *returnType*←functionName(*argumentTypes*). Leaf nodes have no arguments.

the sun (Hölldobler and Wilson, 1990). However, we argue thzat most current robotics applications are still far from that level of sophistication. Moreover, the discovery of pure-pheromone behaviors is appealing in that its analysis seems more likely to lead to useful applications of pheromone-like global memories to problems for which such "hand-coded" hacks are of less utility. Last, by using *only* pheromone functions, we hope to move towards a formal description of the system as a variation of dynamic programming.

The work in this paper is concerned with learning foraging behaviors that can find food and nest locations in relatively simple, obstacle-free environments. In an accompanying paper (Panait and Luke, 2004a), we present a hard-coded ant foraging algorithm for environments with obstacles.

Evolving Foraging Strategies

To evolve ant behaviors, we used a form of EC known as "strongly-typed" genetic programming (GP) (Koza, 1992; Montana, 1995). In the common form of genetic programming, which we adopted, evolutionary individuals (candidate solutions) use a parse tree structure representation. Leaf nodes in the parse tree return various external state values for the ant. Internal nodes in the tree are passed by their children and return the result of some function applied to those values. Crossover swaps subtrees among individuals. In strongly-typed GP, type constraints specify which nodes may be children to various other nodes: we used strong typing to enable a large set of available functions operating on vector, scalar, and directional information. Even so, the representational complexity available to the GP learner was significantly less than that afforded in the hand-coded design presented in our accompanying poster paper.

The GP system uses three data types. The first data type is *scalar*, representing any real valued information (for example, the level of food pheromones at the current location). A second data type is *vector*, representing a collection of related scalar values (such as the food pheromone levels in neighboring locations). The third data type is *orientation*, used by ants to decide to which neighboring location (of the possible eight) to move next.

In our approach, a GP individual consists of two trees: the pheromone-depositing tree and the behavior-selection tree. An ant is in one of two states: either he is laden with food, or he is not. The pheromone-depositing tree tells the ant

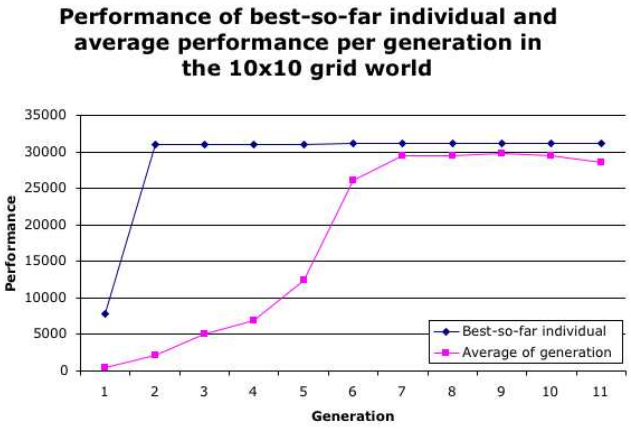

Figure 1: Evolution of performance in the 10x10 grid world.

Figure 2: The emergent foraging pattern for `LearnedBehavior10x10`. Circles represent ants, and squares represent the locations of the nest and the food source.

`LearnedBehavior10x10`

If carrying a food item
  Set the amount of food pheromones to *MaxDistanceFromSite* − *DistanceFromSite*
  Move to the neighboring location with most home pheromones
Else
  Set the amount of home pheromones to *MaxDistanceFromSite* − *DistanceFromSite*
  Move to the neighboring location with most food pheromones

Figure 3: The learned behavior in the 10x10 environment.

how much pheromone to deposit; but the ant's state tells it *which* pheromone to deposit. Additionally, the trees consist of nodes labeled by a given pheromone name (for example, *MaxLocalHomePheromone* for the pheromone to the "home", or nest). These labels are correct when the ant is *not* laden with food, but when the ant has food, the pheromones actually dealt with by these nodes are swapped[1]. Thus for example, when the ant is laden with food *MaxLocalHomePheromone* actually returns the max value of the local pheromone to the *food*, and not the nest.

The root node of the pheromone-depositing tree returns a scalar value (the pheromone to deposit); the absolute value is always used. The root node of the behavior-selection tree returns a direction. Accordingly, these two trees are constructed out of two different sets of nodes. The sets are shown in Table 1. The functions shown are admittedly simple; but for a first attempt we felt this was reasonable.

The algorithm the learning ants followed is:

`Foraging-Behavior`

Call the first tree to select the desired level of pheromones
Call the second tree to select where to move next
Deposit pheromones and move to desired location

The experiments were implemented using the MASON (Luke et al., 2003) multi-agent simulator and the ECJ (Luke, 2002) evolutionary computation framework. The parameters for the EC system were: elitism of size 2, 100 individuals per population, minimum/maximum depth for Ramped Half-and-Half tree generation of 2/4, minimum/maximum depth for Grow tree generation of 3/3, and re-attempting unsuccessful crossover operations 100 times before giving up and returning the original parents. All other parameters have default values as specified in (Koza, 1992). The fitness of an individual is computed as the average performance of three trials. The performance in each trial is calculated as $FoodPickedUp + 10 * FoodCarriedToNest$. The parameters for the multiagent foraging simulation are: minimum/maximum amount of a given type of pheromone per location of 0/100, evaporation rate of 0.1%, and diffusion rate of 0.1%. Demanding simulations resulted in an extremely slow evolutionary process, which limited the cur-

[1]We used this symmetry in the foraging task to reduce the size of the search space.

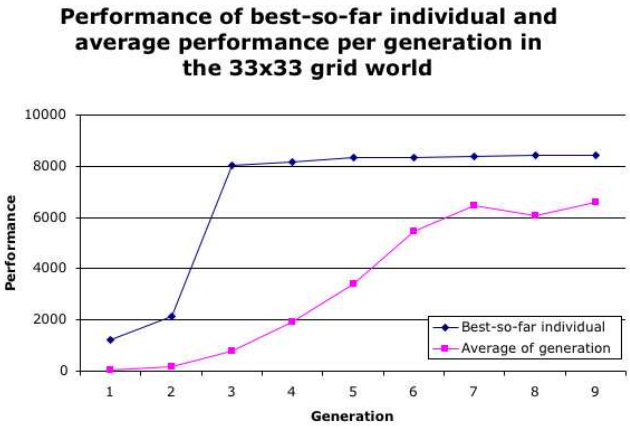

Figure 4: Evolution of performance in the 33x33 grid world.

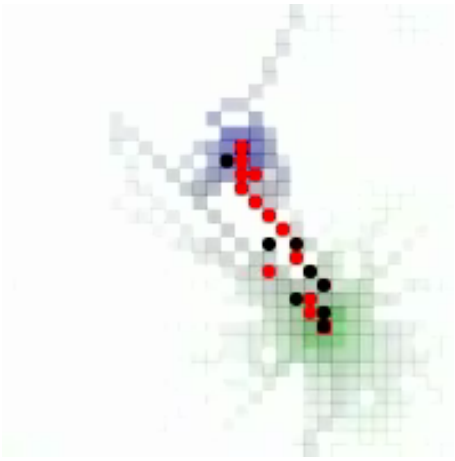

Figure 5: The emergent foraging pattern for `LearnedBehavior33x33`. Dots represent ants, and shades of grey indicate concentration of pheromones.

`LearnedBehavior33x33`
If carrying a food item
Set the amount of food pheromones to *MaxPheromoneDividedByDistanceFromSite*
Move to the neighboring location with minimum value for $FoodPheromones - 3 * HomePheromones$
Else
Set the amount of home pheromones to *MaxPheromoneDividedByDistanceFromSite*
Move to the neighboring location with minimum value for $HomePheromones - 3 * FoodPheromones$

Figure 6: The learned behavior in the 33x33 environment.

rent experiment to only a single run consisting of few generations. Additional runs are required to make statistically significant conclusions; so this work should be considered proof-of-concept only.

Likewise, the proof-of-concept experimentation presented in this paper relies on three assumptions primarily as a simplification of the search space. Other work of ours (Panait and Luke, 2004a,b) has eliminated these assumptions for hand-coded ant algorithms, and we have no reason to doubt that such elimination would be problematic for an evolutionary method in future work. The first assumption is that the agents can move to any of the eight neighboring locations (this eliminates obstacles and requires the world to be toroidal). Second, ants die and new ants are created at the nest. Third, ants cannot only add, but can also *remove* pheromones from the environment (the concept of anti-pheromones was previously used in (Montgomery and Randall, 2002) to improve exploration and help the system escape from local optima).

Experiments

The first experiment concerned learning foraging behaviors in a small 10x10 toroidal grid world. Other parameters for the ant foraging simulation were as follows: 501 simulation steps, food source located at (5,3), nest located at (7,7), ant lifespan of 50 simulation steps, one initial ant, one new ant per time step, and maximum 50 ants in simulation at each time step.

The performance of the best-so-far individuals and the average performance per generation are plotted in Figure 1. The graph shows that a good solution is discovered relatively easily, within two generations. The behavior of a well performing forager, as well as an emergent foraging trail it creates, are shown in Figures 2 and 3.

The second experiment concerned learning foraging behaviors in a larger 33x33 grid world. Other parameters for the ant foraging simulation were as follows: 1001 simulation steps, food source located at (17,10), nest located at (23,23), ant lifespan of 50 simulation steps, one initial ant, one new ant per time step, and maximum 50 ants in simulation at each time step.

The performance of the best-so-far individuals and the average performance per generation are plotted in Figure 4. The graph shows that a good solution is still discovered relatively easily, within three generations. The new individual contains a simple, but useful formula for exploring: when

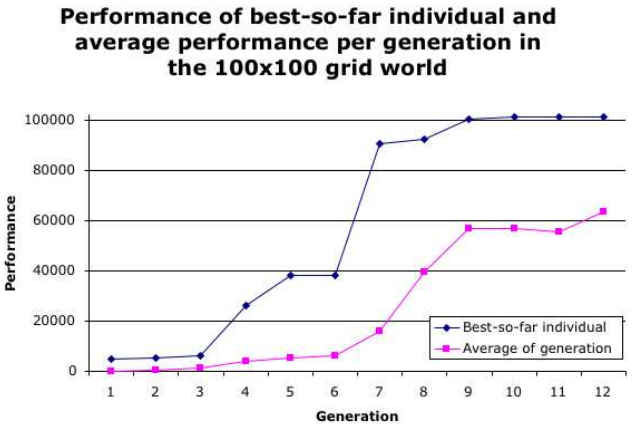

Figure 7: Evolution of performance in the 100x100 grid world.

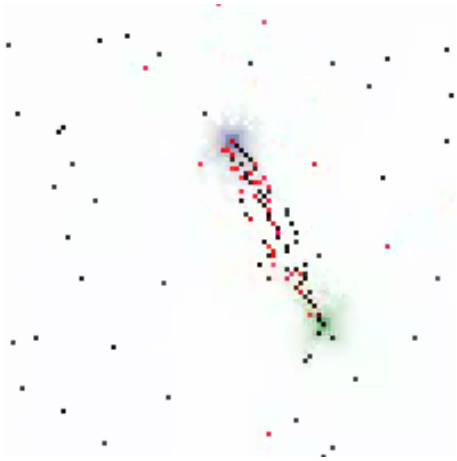

Figure 8: The emergent foraging pattern for `LearnedBehavior100x100`. Dots represent ants, and shades of grey indicate concentration of pheromones.

`LearnedBehavior100x100`

If carrying a food item

Set the amount of food pheromones to *Max(MinLocalFoodPheromone, MaxPheromoneDividedByDistanceFromSite)*

Move to the neighboring location with minimum value for $FoodPheromones - 2 * HomePheromones$

Else

Set the amount of home pheromones to *Min(MinLocalHomePheromone, MaxPheromoneDividedByDistanceFromSite)*

Move to the neighboring location with minimum value for $HomePheromones - 2 * FoodPheromones$

Figure 9: The learned behavior in the 100x100 environment.

searching for the food source, advance towards more food pheromones and also less nest pheromones. This improves the initial search process by guiding the ants away from the nest, and it represents an interesting alternative to the exploration strategy in our accompanying paper (Panait and Luke, 2004a). A behavior of a well performing ant forager is shown in Figure 6, and an emergent foraging trail in one application of the specific learned foraging behavior is presented in Figure 5.

The third experiment concerned learning foraging behaviors in a 100x100 grid world. Other parameters for the ant foraging simulation were as follows: 2501 simulation steps, food source located at (50,30), nest located at (70,70), ant lifespan of 500 simulation steps, one initial ant, one new ant per time step, and maximum 500 ants in simulation at each time step.

The performance of the best-so-far individuals and the average performance per generation are plotted in Figure 7. The graph shows continual improvement over the first nine generations, suggesting incrementally more complex foraging strategies are discovered. A relatively similar, but somewhat more complex individual was discovered; its behavioral algorithm is presented in Figure 9. Additionally, Figure 8 presents an emergent foraging trail obtained when using this behavior for the ants, and it shows how most of the ants have converged on a straight trail connecting the nest to the food source.

In the fourth experiment, we took the best evolved individuals from each of the three previous experiments and tested them in all three environments. For each individual and each grid size, we performed 10 runs. The results are shown in Table 2.

As can be seen, the more difficult the training problem domain, the more general the solutions (they perform equally well to other solutions specifically evolved for simpler domains). This suggests that for simpler problems, there is no learning gradient toward more sophisticated foraging behaviors. Rather, simple enough such strategies perform as well as more advanced strategies, and the learning system is not capable to distinguish among them. However, as the problem domain becomes more and more challenging, increasingly general foraging strategies are discovered. Additional experiments are required to support this hypothesis.

	10x10 environment	33x33 environment	100x100 environment
LearnedBehavior10x10	**2801.00** (27.91)	113.80 (281.78)	2.20 (3.52)
LearnedBehavior33x33	**2800.50** (38.65)	**929.80** (17.67)	3958.50 (2808.00)
LearnedBehavior100x100	**2802.90** (26.77)	**931.90** (18.10)	**7098.90** (1636.22)

Table 2: The performance of evolved foraging behaviors across the three grid worlds. Numbers (mean performance, standard deviation in parentheses) summarize food items returned to the nest in 10 runs. Bold numbers represent statistically significantly better performance (95% confidence) for a given grid size (down a column).

Conclusions and Future Work

This paper presented a successful application of evolutionary learning to search the space of foraging behaviors. The results of the learning approach are agent behaviors capable of learning by themselves: they use pheromones to mark trails connecting the nest to the food source such that they can navigate faster between the two sites.

Additionally, the results suggest that behaviors learned in more difficult domains also have good performance in easier ones. However, the opposite does not hold: behaviors learned with for easier settings have significantly worse performance when tested on more difficult problems.

Our future work will analyze representational bias, eliminate some of the simplifying assumptions in our model, and apply the learning approach to more complex domains possibly involving obstacles, multiple possibly-decaying food sources, and predator agents, all of which may require the system to develop specialized behaviors for different ants.

Acknowledgments

The authors would like to thank Elena Popovici, Gabriel Balan, Zbigniew Skolicki, Jeff Bassett, Paul Wiegand and Marcel Barbulescu for discussions and suggestions related to the ant algorithms.

References

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press.

Collins, R. J. and Jefferson, D. R. (1992). Antfarm : Towards simulated evolution. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, pages 579–601. Addison-Wesley, Redwood City, CA.

Hölldobler, B. and Wilson, E. O. (1990). *The Ants*. Harvard University Press.

Koza, J. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press.

Leerink, L. R., Schultz, S. R., and Jabri, M. A. (1995). A reinforcement learning exploration strategy based on ant foraging mechanisms. In *Proceedings of the Sixth Australian Conference on Neural Networks*, Sydney, Australia.

Luke, S. (2002). ECJ 9: A Java EC research system. http://www.cs.umd.edu/projects/plus/ec/ecj/.

Luke, S., Balan, G. C., Panait, L. A., Cioffi-Revilla, C., and Paus, S. (2003). MASON: A Java multi-agent simulation library. In *Proceedings of Agent 2003 Conference on Challenges in Social Simulation*.

Monekosso, N., Remagnino, P., and Szarowicz, A. (2002). An improved q-learning algorithm using synthetic pheromones. In B. Dunin-Keplicz, E. N., editor, *From Theory to Practice in Multi-Agent Systems*, Lecture Notes in Artificial Intelligence LNAI-2296. Springer-Verlag.

Montana, D. J. (1995). Strongly typed genetic programming. *Evolutionary Computation*, 3:199–230.

Montgomery, J. and Randall, M. (2002). Anti-Pheromone as a Tool for Better Exploration of Search Spaces. In et al, M. D., editor, *Ant Algorithms: Third International Workshop (ANTS 2002)*, Lecture Notes in Computer Science LNCS 2463. Springer-Verlag.

Panait, L. and Luke, S. (2004a). Ant foraging revisited. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*.

Panait, L. and Luke, S. (2004b). A pheromone-based utility model for collaborative foraging. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-2004)*.

Sauter, J., Matthews, R. S., Parunak, H. V. D., and Brueckner, S. (2002). Evolving adaptive pheromone path planning mechanisms. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 434–440.

Sudd, J. H. and Franks, N. R. (1987). *The Behavioral Ecology of Ants*. Chapman & Hall, New York.

White, T., Pagurek, B., and Oppacher, F. (1998). ASGA : Improving the ant system by integration with genetic algorithms. In et al, J. R. K., editor, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 610–617. Morgan Kaufmann.