# Evolving an Ecology of Two-Tiered Organizations

Travis L. Kriplean
University of Wisconsin-Madison
kriplean@cs.wisc.edu

## ABSTRACT

Evolutionary models typically rely on a single level of evolution for training a team of cooperating agents. I present a model that evolves at two levels—an "organizational" level and the more traditional "individual" level. Each organization contains an embedded agent population that goes through a full evolutionary process every organizational time-step. The organization's genetic code is essentially a policy that specifies the training process for its embedded agents. It also defines the creation of a representative team that is compiled after each organizational time-step. An organization's fitness is based on the performance of this representative team.

## Categories and Subject Descriptors

I.2.8-Problem Solving, Control Methods and Search

## General Terms

Algorithms, Design

## Keywords

Evolutionary computation, genetic programming, genetic algorithms, distributed artificial intelligence, teamwork

## 1. INTRODUCTION

When evolving a team of coordinating agents or co-adapted subcomponents, researchers tend to focus on evolution at the individual level. Because organizations have a life of their own [7], the emphasis on a single level ignores the potential of an explicit organization.

In this paper, I describe a two-tiered model that evolves a population of "organizations", each of which contains a nested population of individual agents. The term *agent* will be used for the individuals within an organization. *Tier* and *level* are used interchangeably to refer to the two evolutionary stages.

Each organization specifies, through its genetic code, multiple sequences of training tasks that its embedded agents go through at each time-step. After training the agents, the organization assembles the best into a team. The performance of this team on the domain problem determines the organization's fitness.


*Genetic and Evolutionary Computation Conference (GECCO)'05*, June 25-29, 2005, Washington, DC, USA.


The underlying motivation for this model is to create a stronger framework for agent-based modeling in the social sciences [1]. Explicit representation of organizational processes will allow richer social dynamics to be captured. For distributed artificial intelligence, the two-tiered model supports the emergence of coordination in multi-agent systems where there may be affinities between agents trained differently. For example, in soccer, training only for goalies, forwards, or sweepers would result in a poor team. But with a heterogeneous team, a better outcome is expected.

In the two-tiered model, the organization defines agent-roles and is responsible for creating a team. In this way, the organizational level is a parallelized search of the space of possible teams, distinct from the Punctuated Anytime Learning model [3], where the learning system, for the whole population, is periodically adjusted during the evolutionary process.

The model developed here is not oriented to a specific problem domain. It should work best in decomposable domains where the most successful training process is unclear and the task requires heterogeneous agents contributing different skill sets [8, 10]. The model allows the emergence of unique systems of agent coordination that operate successfully only within the context of the organization's specific policies.

The study will be organized in three descending grades of abstraction: First I describe the two-tiered model in its most abstract form: the organizational ontology, relationship between the two levels of evolution, and creation of a team. Next, a domain-independent implementation is described. Here genetic algorithms represent organizations and genetic programs represent individual agents. I will then briefly describe the model's application in a predator-prey model. The last section points to future work that may improve the current two-tiered model.

## 2. THE TWO-TIERED MODEL

The two-tiered model consists of nested evolutionary populations—organizations that in turn contain a population of individuals. The central idea is to let the organizations evolve training strategies for their agents and then compile an elite team for testing on the domain problem.

### 2.1 The Evolutionary Scheme

A time-step at the organizational level completely encapsulates a full training regime at the individual level. Individuals are maintained *across* organizational generations—there is no agent turnover after each organizational time-step. Thus, the organization may change while maintaining its members. This decision is based on sociological research suggesting that intraorganizational dynamics are better modeled by Lamarckian

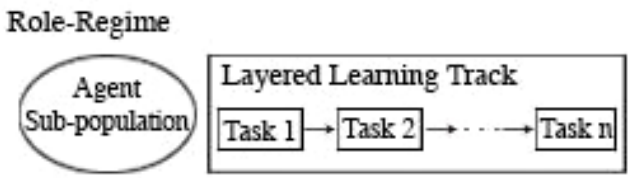
processes [4] and has the added benefit of computational efficiency, because agent programs don't have to be retrained from scratch every generation.

## 2.2 Reifying the Organization

The organization becomes important by partitioning its embedded agents into sub-populations and specifying sequences of sub-problems for each to train on at each organizational time-step. The two-tiered model draws on *layered learning* (LL) to assist in this training process.

It has been shown that LL improves the training process in domains decomposable into hierarchically structured sub-problems [5, 14, 15]. In LL, a population of agents is trained on a series of sub-problems before confronting the domain problem. The sub-problems themselves are user-defined (domain decomposition is not machine learned [14]) and should be relevant to "solving" the domain problem.

A few terms need to be introduced at this point. Defined sub-problems will be referred to as *tasks.* For example, tasks relevant to soccer might include dribbling, passing, defending, and shooting. The sequence of tasks used to train a population of agents is called a *layered learning track* (LL-track). I call the association of agents with a LL-track a *role regime* (RR)*,* because training on a LL-track prepares its agents to assume a role (figure 1)*.* Drawing on soccer again, agents trained on the LL-track dribble-pass-shoot might reasonably be considered learning the agent-role "forward."



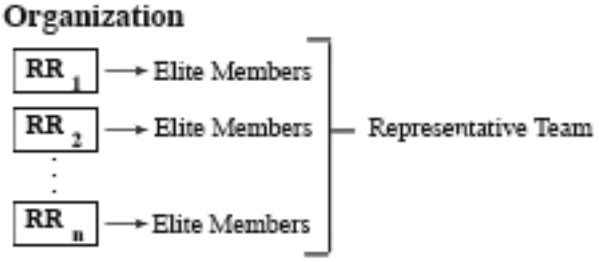

An essential responsibility of the organization is to specify the RR by defining LL-tracks from an available set of tasks. If there were only a single RR, then the agents would be homogenous with respect to training, a weakness in complex domains. Thus, in the two-tiered model, each organization establishes multiple RRs. The organization's definition of the RRs, and the subsequent administration of this training policy, is called the *training regime*.

Using multiple RRs is similar to the idea of *mixed increments*, where a set of populations trained on different tasks are combined to form a team [16]. However, the mixed-increments approach doesn't construct sequences of tasks to be trained on.

To summarize, the training regime, defined in the organization's genetic code, specifies multiple RRs, each containing agents trained on a LL-track.

## 2.3 Team Creation and Organization Fitness

At every evolutionary step, each organization needs to be evaluated. As the two-tiered model is meant to evolve successful teamwork, the fitness of an organization is based on the performance of a *representative team* drawn from its embedded agents. After each organization administers its training regime, the organization compiles a team by drawing on the best individuals from each RR, where the number of agents drawn from each is specified in the organization's genetic code and is bounded by a maximum team size (figure 2).




After the organization has assembled its representative tour de force, it needs to be tested on the domain problem itself. *This test entails a final evolutionary training process in and of itself.* The evaluation system's population is seeded with multiple clones of the star team. Then a team-based evolutionary process takes place, where inter-role coordination may be learned. The organization's fitness is based upon the quality of the final generation of these teams.

# 3. IMPLEMENTATION

In this section I present a domain-independent implementation of the two-tiered model. Genetic algorithms (GA) represent organizations and genetic programs (GP) represent the individual agent. The system is built on Breve, a 3d simulation environment for artificial life investigations [6]. I use the Push programming language to represent the individual agents' genetic code. Push is a Lisp-like, stack-based language designed specifically for evolutionary computation systems [12, 13]. Breve interfaces well with Push.

## 3.1 Representations

The particular evolutionary algorithms used at each level of the two-tiered model are described in this section. At the individual level, agents are modeled as genetic programs. GP is an attractive choice for the individual because of the flexibility that it gives per domain.

The organization is represented as a GA. A GA is an apt data structure for the organization because it is essentially a policy that negotiates the construction and the training of a team (Table 1).

**Table 1. An Organization's Genetic Code (* means >0)**

| | |
|---|---|
| Organization | := (RR)* |
| RR | := ((NumAgents) & (LL-Track)) |
| LL-Track | := (Task)* |
| Task | := pre-defined task identifier |
| NumAgents | := agents to use from this RR |

Evolution at the organizational level is a search of the space of possible teams. The size of this space is:

$$2^{TotalRR(\lceil \log_2 TeamSize \rceil + TasksPerRR \times \lceil \log_2 AvailableTasks \rceil)}$$

## 3.2 Evaluation of the Organization

This section addresses the details of the evolutionary process that takes place after an organization administers its training regime and a representative team is selected. The result of this process determines the fitness of the organization. This evolutionary stage starts with a population seeded by clones of the organization's representative team and trains on the full domain problem.

In this team-based evolutionary stage, crossover is restricted to members drawn from the same RR. Because all the agents, no matter which RR they belong to, have access to the same function set, the restriction is made only to protect the integrity of roles [8]. Each RR might be viewed as its own species because of this breeding restriction,.

## 3.3 GA Considerations for the Organization

If standard GA techniques were used in the reproduction stage at the organizational level, each organization would be severed from its embedded agents as each new generation of organizations is created. Because one of the underlying ideas of this model is that agents adapt to their organizations, even as the organizations themselves change, these standard GA techniques are too disruptive for the agents.

I use what I'll call a *stable GA*. Here, every organization's genetic code is guaranteed to be represented in the next generation to some extent, coupled with the same agents. For every organization, its genetic code is subject to standard GA operators like mutation and crossover, with a chance for direct copying. Other unique genetic operators, such as task-exchange and RR-exchange, are implemented. Aside from being less disruptive, the stable GA maintains greater organizational diversity, at the expense of focused exploitation of rich areas of the organization-fitness landscape.

# 4. PRELIMINARY RESULTS

In this section, I describe an implementation of the two-tiered model for a classic predator-prey model, the pursuit problem. This section should be read as a clarification of the concepts underlying the two-tiered model rather than a demonstration of its potency. The results for the current implementation are inconclusive for reasons described later.

## 4.1 The Pursuit Problem

The pursuit domain is popular for testing new techniques in distributed artificial intelligence. In the classic pursuit problem [2], four predators try to capture one prey. The environment is a grid and capture occurs when the predators surround the prey on all four sides. Each square of the grid can only be occupied by a single agent and movement is constrained to up, down, left, and right. The prey is generally slower than the predators, moves randomly, and probabilistically stands still.

Because of the freedom of Breve, I elected to implement the domain in a different fashion: the geography is continuous, agents move simultaneously, and capture is defined as any agent touching the prey (which moves faster than the predators). The prey reacts to the predators by actively avoiding them, with a 50-50 chance of moving directly away from the nearest predator or towards the nearest edge at a 45° angle with the directly-away-from-predator vector, but only if the nearest predator is within a certain radius.

## 4.2 Description of Agent Function Set

The predators, as genetic programs, have access to both basic operations and domain-specific functions. The basic operations include standard Boolean functions, float operations, and vector operations (cross- and dot-product, add, subtract, divide, and multiply). Additionally, a function outputting the angle between two vectors is available. Agents can query for their own position and velocity. The most important domain-specific functions are vector-to-prey and orthogonal-vector-to-prey.

Each predator can refer to one another by name (*name-based sensing*) as well as relationally (*deictic-sensing*, e.g. "nearest-agent") [8]. Using the name-based sensing functions, predators can query for another predator's location or velocity. Using the deictic-sensing functions, predators can query for the nearest predator based on the following criteria: overall, to the right, left, front, and back. These are calculated after the world-axis is rotated with respect to the predator's direction vector so that "facing" has meaning. In the future, *role-based deictic sensing* will be enabled, where each predator is able to deictically sense according to a specified RR.

On every simulation iteration, after the prey behaves, each predator's program is executed. The output of the program determines the predator's behavior—the top of the vector stack is the predator's direction and the top of the float stack is the predator's speed, capped at a maximum.

## 4.3 Available Tasks for Pursuit

Organizations were allowed to construct three RRs with three tasks each. The available tasks include "approach", "circle", "spiral", and "spread". In "approach", a predator's fitness is based on final proximity to the prey, with maximum fitness assigned to predators that touch the prey. "Circle" uses a predator's positional history to calculate a fitness based on each successive angle change (relative to the prey), where changes in the distance to the prey are penalized. "Spiral" is basically the same as "circle", except that proximity to the prey is rewarded. The last task is called "spread"—this is a team-based task that penalizes deviations from equal angle spacing (in relation to the prey) between each predator in the final configuration.

## 4.4 Evaluation of the Organization

In this implementation of the pursuit problem, during the evolutionary process that results in a fitness assignment to the organization, every team gets a chance to capture the prey as many times as possible within a given time frame. If the team successfully captures the prey, positions are reset and the team can capture it again before time expires. The fitness of an organization is the total prey captures during the last generation.

## 4.5 Results

In my initial tests in the pursuit domain, the two-tiered model performs about the same as scratch-GP (single level GP without any task-decomposition). This is most likely due to the current implementation of the mutation operator. The mutation operator adds a new random subtree to the program instead of probabilistically morphing each expression in the program.

This version of the mutation operator takes the potency out of LL. LL depends on each task being relevant to the next one in the sequence, but this mutation operator hampers the hierarchal sequencing of tasks. This leads to the domination of an easy-to-find local optimum, the charge-the-prey-no-matter-what strategy. Constraining the number of tasks per RR to one (simulating mixed-increments [16]) or the number of RRs to one, with three tasks in the RR (a version of LL), also leads to performance equivalent to scratch-GP, even though these strategies have been shown to be successful in other domains [14, 15, 16].

A drawback of the two-tiered model, typical of explicit evolution of heterogeneous systems [10], is simply computational effort — evolution on two levels, especially with embedded GP systems, is very computationally intensive.

# 5. FUTURE EXTENSIONS

There are many improvements that can be made to the current two-tiered model. This section presents the most promising. First, there is no inter-agent communication right now. Despite some evidence that shows communication isn't always necessary [11], I see this as a serious block to the evolution of teamwork. A blackboard should be added for agents to communicate with one another.

Another potential improvement aims at mitigating the credit-assignment problem, a recurring issue in team-based evolution. I chose to blanket fitness assignment in team-based evolution because, in heterogeneous systems, where an agent might perform an essential role, yet whose behavior cannot be objectively measured, assigning fitness at an individual level is harmful and unfair. However, I witnessed the credit-assignment problem frequently, where freeloaders were evaluated just as favorably as the predators that actually made the prey react. In order to try to deal with this problem, bonus points might be given to agents that impact the situation in some way. For instance, in the pursuit domain, those predators that make the prey react would get more credit than those that didn't. While these reactive behaviors may be detrimental to the goals of the team, giving them credit will reduce the number of agents who don't do anything even remotely connected to the task.

A third extension centers on the type of trainable teamwork the organization can specify. The only chance (before training on the domain problem) that agents get to develop teamwork is within each RR; this is to say, the current model only supports intra-role training of teamwork. Only after the final team has been selected and the organization is being evaluated do agents of different roles interact. However, heterogeneous teamwork requires solid inter-role coordination, where helpful decompositions of the domain problem might involve agents of different roles. A major extension of the two-tiered model would be to add an inter-role training layer, where an organization might specify training tasks that combine multiple roles. Another approach would borrow an idea from *cooperative coevolution* [9], where intra-role training occurs in the context of non-evolving stand-in representatives of other roles. Thus, although the role evolution is self-contained, the agents would have the chance to adapt to agents of other roles.

A final extension would allow the organization to create its own tasks by selecting from an available set of task templates and generating a fitness function based on an arbitrary composition of available fitness functions. For an example, a specification of the "spiral" task may be accomplished by using a template that records positional history and then composing a fitness function based on angular change per step and distance to prey.

# 6. CONCLUSION

Although results in the pursuit domain are inconclusive, the two-tiered model has the potential to provide evolutionary systems with greater traction in complex domains, where unknown combinations of heterogeneous agents perform well.

# 7. ACKNOWLEDGMENTS

I'd like to thank my advisor, Professor Robert R. Meyer, for the time, help and direction he's provided and Jon Klein for his quick responses to Breve implementation requests. I'd also like to thank Anton Vaynshtok, Fred Moore, Beth Schewe, and three anonymous reviewers for helpful comments on earlier drafts.

# 8. REFERENCES

[1] Axelrod, R. *The Complexity of Cooperation.* Princeton University Press, Princeton, NJ, 1997.

[2] Benda, M., Jagannathan, V., and Dodhiawalla, R. *On Optimal Cooperation of Knowledge Sources.* Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA, August 1985.

[3] Blumenthal, H.J. and Parker, G.B. Co-Evolving Team Capture Strategies for Dissimilar Robots. In *AAAI Fall Symposium* (Arlington, Virginia, Oct. 21-24, 2004). AAAI Press, 15-23.

[4] Bryce, D. and Singh, J. The Future of the Firm from an Evolutionary Perspective. In DiMaggio, Paul (ed.) *The 21st Century Firm: Changing Economic Organization in International Perspective.* Princeton University Press, Princeton, NJ, 2001, 160-85.

[5] Hsu, W. and Gustafson, S. M. Genetic Programming and Multi-Agent Layered Learning by Reinforcements. In *Proceedings of the Genetic and Evolutionary Computation Conference*. (New York, USA, July 9-13, 2002). Morgan Kaufmann Publishers, New York, 2002, 764-771.

[6] Klein, J. BREVE: A 3d Simulation Environment for the Simulation of Decentralized Systems and Artificial Life. *Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems*. (U. of New South Wales, Sydney, Australia, Dec. 9-13, 2002). MIT Press, Cambridge, MA, 2002, 329-35.

[7] Levitt, B. and March, J. Organizational Learning. In *Annual Review of Sociology,* 14 (1988), 319-40.

[8] Luke, S. and Spector, L. Evolving Teamwork and Coordination with Genetic Programming. In *Genetic*

*Programming 1996: Proceedings of the First Annual Conference*. (Stanford University, CA, July 28-31, 1996). MIT Press, Cambridge, MA. 1996, 141-9.

[9] Potter, M. A. and De Jong, K.A. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. In *Evolutionary Computation*, 8(1), 2000, 1-29.

[10] Potter, M.A., Meeden, L.A., and Schultz, A.C. Heterogeneity in the Coevolved Behaviors of Mobile Robots: The Emergence of Specialists. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*. (Seattle, WA, Aug. 4-10, 2001). Morgan Kaufmann Publishers, San Francisco, CA, 1337-43.

[11] Sen, S., Sekaran, M., and Hale, J. Learning To Coordinate Without Sharing Information. In *Proceedings of the National Conference on Artificial Intelligence*. (Seattle, WA, July 31-August 4, 1994). AAAI Press, 426-31.

[12] Spector, L., Klein, J., Perry, C., and Feinstein, M. Emergence of Collective Behavior in Evolving Populations of Flying Agents. In *Proceedings of the Genetic and Evolutionary Computation Conference.* (Chicago, IL, July 12-16, 2003). Springer-Verlag, Berline, 2003, 61-73.

[13] Spector, L., Perry, C., Klein, J., and Keijzer, M. Push 3.0 Programming Language Description. http://hampshire.edu/lspector/push3-description.html, 2004.

[14] Stone, P. and Veloso, M. Layered Learning. In *Proceedings of the Eleventh European Conference on Machine Learning*. (Barcelona, Catalonia, Spain, May 30-June 2, 2000). Springer Verlag, 369-381.

[15] ----Multiagent Systems: A Survey from a Machine Learning Perspective. In *Autonomous Robotics*, 8 (3). July 2000.

[16] Winkeler, J. and Manjunath, B.S. Incremental Evolution in Genetic Programming. In *Proceedings of the Third Annual Conference on Genetic Programming.* (U. of Wisconsin, July 22-25, 1998). Morgan Kaufmann Publishers, San Francisco, CA, 403-411.