Tags and the Evolution of Cooperation in Complex Environments

Lee Spector[1], **Jon Klein**[1,2] and **Chris Perry**[1]

[1]Hampshire College
Amherst, MA 01002
[2]Physical Resource Theory, Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
lspector@hampshire.edu

Abstract

Cooperation in evolving populations of agents has been explained as arising from kin selection, reciprocity during repeated interactions, and indirect reciprocity through agent reputations. All of these mechanisms require significant agent capabilities, but recent research using computational models has shown that arbitrary markers called "tags" can be used to achieve significant levels of cooperation even in the absence of memory, repeated interactions or knowledge of kin. This is important because it helps to explain the evolution of cooperation in organisms with limited cognitive capabilities, and also because it may help us to engineer cooperative behaviors in multi-agent systems. The computational models used in previous studies, however, have typically been constrained such that cooperation is the only viable strategy for gaining an evolutionary advantage. Here we show that tag-mediated recognition can lead to significant levels of cooperation in a less constrained artificial life simulation, even when other viable survival strategies exist. The results suggest that tags provide a simple yet effective mechanism for promoting the emergence of collective behaviors in evolving agent populations.

Introduction

Altruistic behaviors among individuals in evolving populations can be attributed to a variety of mechanisms. Kin selection explains how cooperation emerges between closely related individuals (Hamilton, 1963). In repeated interactions, cooperation can emerge due to reciprocity even when defection is most beneficial in the short term (Trivers, 1972; Axelrod and Hamilton, 1981). Even when individuals are not likely to engage in repeated interactions, cooperation can emerge through "indirect reciprocity" based on image scoring and reputation (Nowak and Sigmund, 1998).

More recently, Riolo et al. introduced a model of cooperation in which individuals have no knowledge of kin, no significant repeated interactions, no image scoring and no memory. Instead, they showed that cooperation can arise based on the existence of arbitrary identifying markers called "tags" (Riolo et al., 2001). In their model, each agent was given a tag value, a real number between 0.0 and 1.0, and a tolerance value, also between 0.0 and 1.0. Each agent would cooperate with another (by donating energy) only when its own tag differed from the other's by less than the tolerance value. Riolo et al. showed that significant levels of cooperation could emerge, even though the agents had no repeated interactions and no other recognition mechanisms.

In contrast to the other cooperation mechanisms mentioned above, tags require no memory and little to no cognitive ability. Tag-mediated recognition could occur on the level of single molecules, such as cell surface proteins or pheromones. While tags can be genetically encoded, they do not necessarily connote more general genetic similarity. Indeed, as predicted by Richard Dawkins' "green beard effect" (Dawkins, 1976), cooperation can emerge from a single gene which both triggers the expression of a tag (such as a chemical signal, or a green beard) and triggers altruistic behaviors toward other individuals bearing the same tag. This "green beard effect" has recently been found to be the mechanism behind altruistic behaviors of the social slime mold Dictyostelium discoideum (Queller et al., 2003).

While the green beard effect is a straightforward example of how tag-mediated recognition can give rise to cooperation, the tag itself need not be a gene that triggers altruistic behaviors toward other agents possessing the gene. The tag, as we will show, can be completely independent of the genes that trigger the cooperative behavior.

In the context of evolving multi-agent systems, tags provide a remarkably simple mechanism for en-

abling the evolution of cooperation and other collective behaviors. Cooperative behaviors can emerge when agents are given the simple ability to sense and respond to tags expressed by other agents, and the tag can be as simple as a single real number.

While the model presented by Riolo et al. showed that significant levels of cooperation could be achieved using only a simple tag, it did so in a very constrained environment in which sharing was the only agent behavior modeled. Although an agent need not actively share with others in order to reap the benefits, cooperation is nonetheless the only mechanism by which any agent can attain an evolutionary advantage over others. In addition, the original model used a high (10:1) ratio of benefit to cost; the cost to a cooperating agent was far smaller than the benefit to the recipient of the donation. In this paper, we seek to address these concerns to determine whether tags are a viable mechanism for cooperation in a less constrained environment in which other survival strategies exist and with a lower cost-to-benefit ratio.

We describe the emergence of tag-mediated cooperation in a 3D simulation of flying agents called SwarmEvolveTags. We compare the levels of cooperation achieved using tag-mediated recognition to those achieved using other recognition mechanisms. This paper builds on previous work using our SwarmEvolve 2.0 simulation. Agents in SwarmEvolve 2.0 were capable of a simple self/other recognition using hardcoded tolerance values; SwarmEvolveTags implements a tag-mediated recognition mechanism which closely resembles the model of Riolo et al. and a genetic recognition system. The simulation demonstrates the emergence of tag-mediated cooperation, with a cost to benefit ratio of 1:1, in a more realistic and less constrained environment. SwarmEvolveTags is less constrained than previous models of tag-mediated cooperation in the sense that agents can draw from a far richer repertoire of behaviors — in SwarmEvolveTags, agent behaviors are determined by evolving computer programs and a wide variety of non-cooperative behaviors are viable.

breve and Push

All of the the SwarmEvolve simulations described here were constructed using breve (Klein, 2002), a free, open-source simulation package designed for the rapid construction of decentralized systems and artificial life simulations in 3D worlds. While breve is conceptually similar to simulation frameworks such as Swarm (Minar et al., 1996), it is designed from the ground up for continuous-time simulations in continuous 3D space. breve includes built-in collision detection and response, realistic articulated body physical simulation and a rich OpenGL-based display engine.

Simulations are constructed by defining agent behaviors in a simple object-oriented language called *steve*. The language is intended specifically for use with 3D simulations, providing support for 3D vectors and matrices as native types. Agent behaviors can be further customized via an extensible plugin architecture that allows users to incorporate external libraries or programs into breve simulations.

While breve provides the framework for the simulated world, the evolving agent behaviors of SwarmEvolveTags (and its predecessor, SwarmEvolve 2.0) are implemented using Push, a stack-based language developed by Spector (Spector and Robinson, 2002) for multi-type evolutionary computation. The SwarmEvolveTags simulation presented in this paper uses the Push 2.0 library (Spector et al., 2003b). Push is integrated with breve using a simple plugin built with the C++ Push library. The plugin allows agents in breve to run Push programs to determine their behaviors. The plugin also allows callbacks from Push code into breve so that Push programs can execute *steve* methods to obtain sensor input or to trigger agent behaviors in the simulation. Push allows the dynamic manipulation of code so that agents can develop their own genetic operators instead of relying on hard-coded mutation and crossover operators; see the discussion of *autoconstructive evolution* in the SwarmEvolveTags section below.

SwarmEvolve 1.0 and 2.0

SwarmEvolve is the name for a series of artificial life simulations in which we have investigated the emergence of cooperative and collective behaviors. While environments, behaviors and implementations differ between the different SwarmEvolve experiments, they are all based around the premise of simple flying agents ("birds") navigating a 3D world and competing for limited resources ("feeders").

In SwarmEvolve 1.0 (Spector et al., 2003a; Spector et al., 2004) agent behaviors were determined by an evolving vector of constants in a hard-coded control equation based on Reynolds' classic "boids" flocking algorithm (Reynolds, 1987). Even with such limited agent behaviors, a variety of rich survival strategies emerged, some of which exhib-

ited collective behaviors. The most striking behavior could even be seen as a sort of multicellularity, in which some agents in the group served as protective organs guarding food sources from other species, while others gorged on the food source, serving as digestive organs.

SwarmEvolve 2.0 eliminated both the hardcoded control equations and species distinctions in favor of a less constrained approach. Agent behaviors and species distinctions were determined entirely by evolving Push programs. In this environment, we sought to understand the conditions under which cooperation could emerge by adding instructions that allowed agents to donate their energy to others. While voluntarily reducing one's energy stores would normally appear to be maladaptive, we found that significant levels of food sharing emerged with a variety of settings relating to environmental stability and sharing behaviors.

Additionally, SwarmEvolve 2.0 removed the explicit fitness function used in SwarmEvolve 1.0. SwarmEvolve 1.0 used a GA-like fitness and reproduction scheme in which the fitness of an agent was determined as a function of its age and energy level. Agents with the highest fitness scores were selected for reproduction. In SwarmEvolve 2.0, agents compete for limited resources and control their own reproduction. Evolution is driven by the success of agents that are best suited to these tasks.

SwarmEvolveTags

With SwarmEvolveTags, we seek to investigate further the conditions under which cooperation can emerge by comparing different mechanisms of agent recognition, specifically genetic[1] recognition, in which an agent compares its genetic code to that of others, and tag-mediated recognition, in which a single floating point tag, determined by an agent's program, is used.

As in the other SwarmEvolve experiments, flying agents in SwarmEvolveTags compete for survival in a 3D world. The agents receive boosts of energy, up to a maximum value of 1.0, from spherical energy sources which periodically reposition themselves around the world. Agents lose energy at each timestep due to a small "cost of living." They lose additional energy when they collide with other agents and when they give birth, at which time a portion of the parent agent's energy is transferred to the child. If an agent's energy level drops below 0.0 then the agent dies.

Agent behaviors are represented in the Push language. Each agent has a Push program which is run at every timestep. After executing the program, the top value from the point stack (a 3D vector) is used to set the agent's acceleration vector. The top value from the float stack is used to set the agent's hue, the significance of which is described below.

In addition to standard Push instructions which manipulate built-in Push types, we supply a number of simulation callbacks which allow agents to obtain sensor input and control actuators. The full Push instruction set used in SwarmEvolveTags is shown in Table 1.

In contrast to a GA-style "hand of God" reproduction, agents in SwarmEvolveTags reproduce using *autoconstructive evolution*, in which agents must evolve the ability to reproduce. Using a special "code" stack, along with the code instructions shown in table 1, an agent's program can manipulate its own code (and potentially the code of its neighbors) to produce novel programs. Whenever an agent attempts to produce a child (by executing the Spawn instruction), the top of its code stack is examined. If the expression is empty (which happens rarely once the system has been running for some time) then a newly generated, random program is used for the child. If the expression is not empty then it is used as the child's program, after a possible mutation. Code-manipulation instructions that simplify the construction of standard genetic programming mutation and crossover operators are provided, but agent programs need not use these instructions, and when these instructions are in fact used they may also be combined with other code-manipulation instructions to produce novel genetic operators. To prevent early outbreaks of clones from derailing the evolutionary process a mandatory mutation step is also imposed, although the mutation rate for this step is under the agent's control.[2]

Because agents must evolve the ability to reproduce, it takes some time before a stable population of viable agents emerges. During this period, the population typically dwindles and random agents are continually introduced into the simulation to maintain a user-defined population threshold. The

[1] We differentiate between "kin" recognition which implies explicit knowledge of common ancestry and "genetic" recognition which implies knowledge of genetic similarity with or without common ancestry.

[2] See (Spector et al., 2003a; Spector et al., 2004) for details.

Instructions	Description
Float instructions: +, /, *, -, %, <, >, FROMPOINT *Integer instructions*: +, -, *, /, <, > *Boolean instructions*: AND, OR, NOT *Point (3D vector) instructions*: +, -, /, *, FROM1FLOAT, FROM3FLOATS, CROSS *Code instructions*: DUP, POP, SWAP, QUOTE, DO* ATOM, CAR, CDR, CONS, IF, SIZE, LENGTH	Standard Push instructions
Code instructions: Mutate, Crossover	Potential genetic operators
Spawn	Triggers reproduction
ToFood, FoodIntensity	Information about energy sources
MyAge, MyEnergy, MyHue, MyVelocity MyLocation, MyProgram	Information about self
ToFriend, FriendAge, FriendEnergy, FriendHue, FriendVelocity, FriendLocation, FriendProgram	Information about closest "friend" agent
ToOther, OtherAge, OtherEnergy, OtherHue, OtherVelocity, OtherLocation, OthherProgram	Information about closest "other" agent
FeedFriend, FeedOther	Transfer energy to neighboring agents

Table 1: Push instructions used in SwarmEvolveTags.

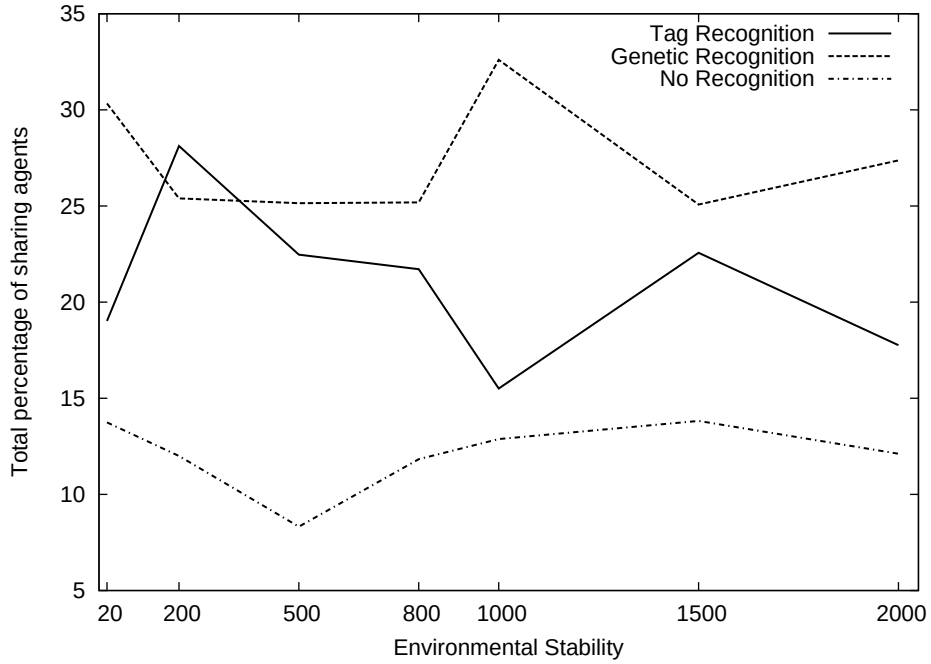

Figure 1: Percentage of successful sharing agents at different environmental stabilities.

agents typically achieve "reproductive competence" after a few hundred iterations and can thereafter maintain a stable population size using their own genetic operators.

Cooperative behavior in SwarmEvolveTags is possible through food sharing: agents can execute two sharing instructions, FeedFriend and FeedOther, which donate food to nearby "friend" and "other" agents, respectively. In order to examine the differences between different recognition mechanisms, we allow for three different modes of friend/other recognition: "genetic recognition," in which the distinction is made based on genetic similarity; "tag-mediated recognition," in which the distinction is made based on tag (hue) similarity; and "none", where all other agents are perceived to be friends.

The tolerance levels for genetic and tag-mediated recognition are determined dynamically by agent programs. When an instruction requiring a friend/other distinction (such as "FeedFriend", or "OtherVelocity") is executed, the top value of the floating point number stack is used as a tolerance value for the comparison. The evolved tolerance values allow agents to exercise a great deal of control over the agents with whom they share and interact. The tolerance values used for recognition need not be statically hardcoded as part of an agent's genome—they may be changed repeatedly at run-time by the agent's program, or even be dynamically computed based on sensor inputs or other calculations.

Genetic recognition is based on differences between Push programs. A "subprogram" in Push is defined as any terminal or sublist found in a program.[3] The difference between two Push programs *p1* and *p2* is defined as the sum of the number of subprograms in *p1* which do not appear in *p2*, and the number of subprograms in *p2* which do not appear in *p1*. This value is normalized by dividing by the sum of the number of subprograms in *p1* and *p2*. Two identical programs thus yield a difference value of 0.0 and two programs which have no subprograms in common have a value of 1.0.

Tag-mediated recognition is based on differences between agents' hue values (on a circular scale that ranges from 0.0 to 1.0) such that two agents with identical hues have a difference of 0.0 and two agents with maximally different hues will have a difference of 0.5. We multiply this value by 2 so that the tag tolerance range matches the genetic tolerance range of 0.0 to 1.0.

[3]Note that this includes the entire program itself; for the purpose of this calculation a program is considered to be a "subprogram" of itself.

Experiment

We ran simulations for each of the three recognition modes at seven different environmental stability settings. The stability of the environment determines the frequency with which an energy source will move to a new location. The probability that an energy source will move at each iteration is given by $\frac{1}{stability}$. The stability settings ranged from 20 (highly unstable) to 2000 (highly stable).

For each of the recognition modes and stability settings, we conducted 65 runs for a total of 1365 runs. Each was allowed to run for 13,000 total iterations, which were divided into 130 epochs of 100 simulation steps. Data collection began only after reproductive competence had been achieved for over 300 iterations and continued for 8,500 iterations. Runs in which reproductive competence was not sustained for at least 8,500 iterations were discarded—this eliminated only 16 runs.

In order to gauge the stability of each sharing strategy, we examined the prevalence of sharing agents that emerged with each sharing mode. We consider a "successful sharing agent" to be an agent that executes a successful sharing action at some point during its life. A successful sharing action is a call to FeedFriend or FeedOther that results in a transfer of energy, meaning that a neighboring agent is found that satisfies the recognition requirement. If no nearby agents are recognized (which might occur, for example, if the active tolerance level is less than 0 for a FeedFriend event), then the sharing event is not recorded. We computed the average number of successful sharing agents by conducting a survey, at each epoch boundary, of all agents was to determine the size of the population and which agents were successful friend- or other-sharers. Population and sharing data were averaged over all runs to compute percentages of sharing vs. non-sharing agents across different recognition modes and stability settings.

Results

Figure 1 shows the percentages of sharing agents at different stability values. Table 2 shows the percentages averaged over all stability levels.

Tag-mediated recognition produces a larger percentage of cooperating agents than is produced when no recognition mechanism is available, and

recognition	% sharing agents
Genetic	27.3%
Tag	21.0%
None	12.1%

Table 2: Percentage of sharing agents averaged over all stability values.

compares reasonably well to the percentage of sharing agents achieved by genetic recognition at several stability values.

The results demonstrate that tag-mediated recognition yields considerable levels of cooperating agents at all levels of environmental stability.

Discussion & Future Work

One explanation for the success of tag-mediated recognition mechanisms is that they provide a heuristic approximation to genetic similarity. This hypothesis could be tested empirically by measuring the correlation between tag similarity and genetic similarity over many runs.

The apparent inverse relationship between the success of genetic recognition vs. tag-mediated recognition at various levels of environmental stability, as visible in Figure 1, is intriguing, but we have no definitive explanation for this. Perhaps this relationship is related to the correlation between tag similarity and genetic similarity.

It would also be interesting to examine the overall level of adaptive sharing that occurs in various conditions. To this end we also examined the total number of sharing *events* (as opposed to sharing *agents*). This data, however, was dominated by the necessarily indiscriminate sharing of the agents in the "no recognition" condition; runs in this condition produced large numbers of sharing events even when there were few sharing agents, and even when the few agents were unsuccessful mutants that left no offspring. A better handle on *adaptive* sharing activity might result from measurement of the number of sharing events produced by agents that meet some threshold of success; such a threshold might be based, for example, on agent lifetime or reproductive success.

Can tags be used to achieve even higher levels of cooperation? While both tag-mediated and genetic recognition can support significant percentages of sharing agents, we observed more sharing (whether measured in terms of sharing events or sharing agents) in the simulations with genetic recognition. This suggests that agents evolved in the context of tag-mediated recognition are pickier about the agents with whom they will share. This persnicketiness may be a product of the fact that agent tags can evolve and change more rapidly and more dramatically than agent genomes, even during a single lifetime. This leads to the conjecture that the use of a static tag over an agent's lifetime will improve the reliability of tags as a recognition mechanism, and will therefore lead to higher levels of sharing in evolving populations of agents that recognize one another by means of tags.

Conclusions

The SwarmEvolveTags simulation results demonstrate that tag-mediated recognition mechanisms can in fact lead to the emergence of significant levels of altruistic behavior between agents, even in unconstrained environments in which other non-cooperative strategies are viable. In addition, this can occur with a cost to benefit ratio of 1:1, a condition that is considerably more challenging than that used in most of the prior research on tag-mediated cooperation.

Although the level of cooperation achieved with tag-mediated recognition is not as high as the level achieved with genetic recognition, the tag-based approach is arguably both simpler and more natural than the current alternatives.. Tag-recognition requires little in the way of cognition or memory, but it may nonetheless form the foundation for cooperative societies of evolving agents.

Source code and screen-shots from the SwarmEvolveTags experiment described in this paper are available online from http://hampshire.edu/lspector/SETags.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0308540 and Grant No. 0216344, and by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30502-00-2-0611.

References

[1] Axelrod, R. and Hamilton, W. D. (1981). The evolution of cooperation. *Science*, 211:1390–1396.

[2] Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press, New York.

[3] Hamilton, W. D. (1963). The evolution of altruistic behavior. *American Naturalist*, 97:354–356.

[4] Klein, J. (2002). breve: a 3d environment for the simulation of decentralized systems and artificial life. In Standish, R., Bedau, M. A., and Abbass, H. A., editors, *Proc. Eighth Intl. Conf. on Artificial Life*, pages 329–334. Cambridge, MA: MIT Press.

[5] Minar, N., Burkhart, R., Langton, C., and Askenazi, M. (1996). The swarm simulation system, a toolkit for building multi-agent simulations. Technical Report 96-06-042, Sante Fe Institute.

[6] Nowak, M. A. and Sigmund, K. (1998). Evolution of indirect reciprocity by image scoring. *Nature*, 393:573–577.

[7] Queller, D. C., Ponte, E., Bozzaro, S., and Strassmann, J. E. (2003). Single-gene greenbeard effects in the social amoeba, dictyostelium discoideum. *Science*, 299:105–106.

[8] Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

[9] Riolo, R. L., Cohen, M. D., and Axelrod, R. (2001). Evolution of cooperation without reciprocity. *Nature*, 414:441–443.

[10] Spector, L., Klein, J., Perry, C., and Feinstein, M. (2003a). Emergence of collective behavior in evolving populations of flying agents. In Cantu-Paz, E., Foster, J. A., Deb, K., Davis, L. D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 61–73. Springer-Verlag.

[11] Spector, L., Klein, J., Perry, C., and Feinstein, M. (2004). Emergence of collective behavior in evolving populations of flying agents. *Genetic Programming and Evolvable Machines*. In press.

[12] Spector, L., Perry, C., and Klein, J. (2003b). Push 2.0 programming language description. Technical report, Hampshire College. `http://hampshire.edu/lspector/push2-description.html`.

[13] Spector, L. and Robinson, A. (2002). Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines*, 3(1):7–40.

[14] Trivers, R. (1972). The evolution of reciprocal altruism. *Quarterly Review of Biology*, 46:35–57.