Evolutionary Design of Particles for Collectively Intelligent Problem Solving

Benjamin Bengfort, Kevin Harrison, and Philip Kim

Abstract—Emergent, self-organizing flocks of particles can be implemented using only simple, local movement behavior rules. When particles are also improved with a limited working memory and goals, the local interactions of these top-down controllers leads to emergent global problem solving such as search and retrieve. We will show in this paper that the rules for both movement and goals can be evolved using Evolutionary Computing techiques on the dynamics parameters, as well as on the finite state machines that control behavior. The evolved particles perform as well as or better then those designed by humans.

1 Introduction

Ever since Reynolds demonstrated that flocks of birds could be simulated as multi-agent systems [13], there has been interest in improving flocks, schools, and swarms to higher levels of collective intelligence. The relatively simple nature of each agent; its motion or goals are determined solely by local interactions; leads to emergent global behavior. Decentralized control would mean optimal and economic solutions to problems in both algorithmic and physical domains. Furthermore, there has been interest in extending such systems to solve more general problems than traditional optimization [9], modeling complex systems [12], [6] or cultural algorithms [5].

This extension has primarily focused on the physical domain to improve multirobot team movement control as in [1], [3], [7], however higher order problem solving is currently being explored. One large area of research is the deployment of mobile robots or sensors for area coverage, either to map terrain, or to establish mesh communication links. In [4], flocking behavior was used to maximize coverage, while still maintaining collective behavior. Other tasks being explored include urban pursuit [17] and robot herding of animals [15].

Many approaches have improved the simple local dynamics by adding minimal, cost effective top down components that also behave locally to produce emergent behavior. One very successful addition to flocks for problem solving is the addition of working memory [18], [8]. Building on flocks with memory improvements, Rodriguez in [14] added goal-driven intelligence in the form of a finite state machine (FSM) that switched the agent between different sets of movement behaviors according to its local environment and current goal. He was able to demonstrate that a team of such agents was able to solve a resource locate-and-collect problem, and that a team programmed with flocking behaviors outperformed teams of agents that did not influence each other's movements.

AMAS theory for designing complex adaptive systems [2] Designing swarms [10]

Evolutionary programming with swarms [16], [11]

In this article, we have attempted to extend Rodriguez's findings by applying evolutionary techniques to the agent control mechanisms. Rodriguez determined the structure of the FSM and the parameters defining each state through inspection and empirical techniques. We attempt to determine the optimal parameters by an evolutionary process that runs a population of randomly-generated FSM configurations through a simulated problem and then evolves the population through fitness-based selection and random mutation.

Our hypothesis was that we could evolve a

configuration for the FSM that would be competitive to a configuration tuned by a human.

2 METHODS

Because we were attempting to demonstrate to an evolutionary process could process an agent controller that was competitive with the hand-tuned controller used by Rodriguez, our experimental setup was intentionally similar to his: two teams of homogenous agents, each with a designated ”home” location, deployed in a two-dimensional world with periodic boundary conditions.

As in Rodriguez, an agent's velocity at each timestep was composed of some combination of the following vectors:

Cohesion: a vector pointing towards the average position of friendly agents in the neighborhood, with magnitude equal to zero when the agent's distance to the average neighbor is zero and increasing quadratically with distance until it is equal to Vmax when the distance to the average neighbor is r.

Alignment: a vector pointing in the same direction as the average velocity of friendly agents in the neighborhood, with magnitude equal to zero when the agent's distance to the neighbors' average position is zero and increasing quadratically with distance until it is equal to Vmax when the distance to the average neighbor is r.

Separation: a vector pointing away from the average position of friendly agents in the neighborhood, with magnitude equal to Vmax when the distance between the agent and the average position is zero, and decreasing quadratically with distance until it is zero when the distance to the average neighbor is r.

Clearance: a vector pointing in a direction orthogonal to the difference between the average neighbor position and the agent's position with magnitude Vmax.

Avoidance: a vector pointing away from an enemy agent in the neighborhood, with magnitude Vmax when distance to the enemy is zero and decreasing linearly with distance until it is zero when the distance is r. Unlike all the other components, this can be applied multiple times each timestep–once for every enemy in the neighborhood.

Also per Rodriguez, the FSM controlling each agent was always in one of four states: searching for new resources, moving to the last known resource deposit, carrying resources back to the home location, or guarding the home or a resource deposit. Each state was composed of some combination of the previously described velocity components, each characterized by a numerical weight and the radius and angle that defined the neighborhood.

Unlike Rodriguez, who prioritized each component and applied them in order, discarding any component that would have caused the velocity to exceed the maximum velocity, velocity in our simulation was a simple linear combination of the components; the maximum velocity constraint was only applied at the end to the resulting sum. Also unlike Rodriguez, agents in our simulation had ”inertia” and started each timestep with the velocity of the previous timestep.

Transitions between states were triggered by conditions in each agent's local environment according to hard-coded rules. For example, an agent in the searching state that detected a mineral deposit within a 200-unit radius (in any direction) would push that deposit onto the top of its memory stack and transition to the seeking state (whcih is characterized by movement toward the top location on its stack).

One of Rodriguez's main findings was that it was advantageous for the agents to post ”guards” on their homes and/or on any discovered resource deposits. Agents that guarded only the home performed best of all, though agents that guarded both the home and deposits still bested non-guarding agents. In order for guarding behavior to be feasible, however, agents must avoid agents on the enemy team. Since we were allowing evolution to determine the strength of this avoidance parameter, we expected that in the absence of a ”natural” motivation for avoidance, the avoidance parameter would be selected down to zero, allowing the agents to ignore enemy guards.

Therefore we implemented a penalty for colliding with another agent. An agent within 10 units of an enemy agent was rendered unable

to move for a number of timesteps equal to 180 minus the angle of incidence in degrees. For example, if an agent collided with an opposing agent at right angles, the agent struck in the side would be unable to move for 90 timesteps, whereas the agent struck in the front would be unable to move for 180 timesteps. Similarly, being "rear-ended" by an opposing agent had no penalty at all. The hope was that by assigning the "responsible" agent more of the penalty, we would create an incentive to avoid collision.

In order to allow for the selection of guarding behavior, the transition to the guarding state was controlled by two evolvable parameters – the "home guarding threshold" and the "deposit guarding threshold" – that specified the number of friendly agents considered sufficient to guard the respective sites; a value of 0 disabled guarding behavior. This was the only instance in our simulation where a transition between states was controlled by an evolvable parameter.

At the beginning of each timestep, therefore, each agent is in a particular state. For each velocity component in its current state, it determines the number of friendly and enemy agents in the respective neighborhood and calculates the direction and magnitude of the component. These components are multiplied by the respective weight, added to the agent's velocity from the previous timestep, and the resulting velocity is subjected to the Vmax constraint. The agent's position is updated by adding the velocity to the agent's position in the previous timestep. The agent then determines whether it should change its state based on the predefined transition rules and its current environment.

3 Experimental Procedure

The computational experiment we ran consisted of pitting two simulated teams in competition with each other in a 3000 x 3000 world for a limited time. We restricted each team to 10 agents, vice 50 in the Rodriguez paper, and used 5 resource deposits instead of 8; both changes were to make the simulation more computationally tractable. The "red" team's behavior was fixed to parameters similar to those found empirically by Rodriguez to yield the best results. At the start of evolution we randomly generated 50 configurations to control the "black" team. Fitness was defined as the number of resources gathered by the black team after 10,000 timesteps.

After running each of the 50 configurations, we subjected the population to a round of evolution. The most fit individual from the current generation was always carried into the next generation, with the other 49 selected via tournaments of size 3. Each of the 49 was then subjected to mutation, with an independent 20% chance of mutating each weight, each radius, and each alpha of every component. If selected for mutation, the trait was changed by a random value uniformly-distributed between -0.2 and 0.2 for weights, -20 and 20 for radii, and -20 and 20 for alphas. Additionally, the home guarding and deposit guarding threshold each had a 20% of changing by plus or minus 1.

4 Results

results

5 Discussion

This work described herein represents only the first steps in the application of evolutionary techniques to this problem domain. By mutating and applying selective pressure to the movement component parameters, we were able to show that a machine could optimize the parameters better than a person. This was a form of optimization, but it did not result in novel behaviors. Indeed, since we hard-wired the transitions between the states (allowing for the guarding threshold to evolve) and allowed the evolver to mutate existing movement components in a given state but not to add them (i.e., we pre-defined which components could have non-zero weight for each state), we held constant the "meaning" of each state.

The next step would be to evolve the structure of the FSM itself; not only to allow the evolver to mix-in any component to any state, but to allow it to add new states, delete old states, and change the transitions between the states.

REFERENCES

[1] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939, 1998.

[2] Davy Capera, JeanYPierre Georgé, M-P Gleizes, and Pierre Glize. The amas theory for complex problem solving based on self-organizing cooperative agents. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, pages 383–388. IEEE, 2003.

[3] Hande Çelikkanat and Erol Şahin. Steering self-organized robot flocks through externally guided individuals. *Neural Computing and Applications*, 19(6):849–865, 2010.

[4] Ke Cheng, Yi Wang, and Prithviraj Dasgupta. Distributed area coverage using robot flocks. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 678–683. IEEE, 2009.

[5] Chan-Jin Chung and Robert G Reynolds. A testbed for solving optimization problems using cultural algorithms. In *Evolutionary programming*, pages 225–236, 1996.

[6] Maurice Clerc and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, 2002.

[7] Jessica Hodgins and David Brogan. Robot herds: Group behaviors for systems with significant dynamics. In *Proceedings of Artificial Life IV*, pages 319–324, 1994.

[8] Xiaohui Hu, Russell C Eberhart, and Yuhui Shi. Particle swarm with extended memory for multiobjective optimization. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 193–197. IEEE, 2003.

[9] James Kennedy, Russell Eberhart, et al. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948. Perth, Australia, 1995.

[10] Maja J Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 432–441, 1993.

[11] Vladimiro Miranda. Evolutionary algorithms with particle swarm movements. In *Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on*, pages 6–21. IEEE, 2005.

[12] Julio M Ottino. Complex systems. *AIChE Journal*, 49(2):292–299, 2003.

[13] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM, 1987.

[14] Alejandro Rodríguez and James A Reggia. Extending self-organizing particle systems to problem solving. *Artificial Life*, 10(4):379–395, 2004.

[15] Richard Vaughan, Neil Sumpter, Jane Henderson, Andy Frost, and Stephen Cameron. Robot control of animal flocks. In *Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings*, pages 277–282. IEEE, 1998.

[16] Chengjian Wei, Zhenya He, Yifeng Zhang, and Wenjiang Pei. Swarm directions embedded in fast evolutionary programming. *networks*, 9:11, 2002.

[17] Ransom Winder and James A Reggia. Using distributed partial memories to improve self-organizing collective movements. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(4):1697–1707, 2004.

[18] Ransom K Winder and James A Reggia. The role of working memory in an urban pursuit scenario. In *Artificial Life*, volume 13, pages 291–298, 2012.