Evolutionary Design of Flocking Particles for Collectively Intelligent Problem Solving

Benjamin Bengfort, Kevin Harrison, and Philip Kim

Abstract—Self-organizing "flocks" of agents that display emergent coordinated behavior can be implemented using only local interaction rules. When agents are also given memory and a goal-based Finite State Machine (FSM) controller, the resulting team is able to solve a locate-and-collect task. We show that Evolutionary Computing techiques can be applied to the FSM controller, where the genotype is a real valued vector representing the FSM and the phenotype is the emergent, self-organizing behavior. Comparing the resulting FSMs to similar controllers described in the previous literature, we find that the evolved FSMs perform as well as or better than those designed by human observation. In this paper we will propose methods for applying evolutionary techniques to the structure and states of the controller, perhaps leading to the emergence of novel behavior.

✦

1 INTRODUCTION

Ever since Reynolds demonstrated that flocks of birds could be simulated as multi-agent systems [16], wherein each agent's motion is determined solely by local interactions with other agents, there has been interest in extending such systems to solve more general problems. The self-organizing characteristic of these systems, whereby coordinated behavior emerges without a central control mechanism, suggests that even relatively simple decentralized systems might yield novel approaches to problems in both the computational and physical domains.

By abstracting the idea of "flocks" to n-dimensional spaces, for example, particle swarms [11], [7] or cultural algorithms [6] have been used for numerical optimization. In the physical domain, swarm behaviors have been used to control multirobot team movement [1], [4], [9], or during deployment of mobile robots or sensors. In [5], flocking behavior was used to maximize area coverage during such deployments coordinating movements via collective behavior. Other tasks using collective problem solving being explored include urban pursuit [20] and robot herding of animals [18].

Many approaches augment the agents with such features as working memory [21], [10]. Our paper builds upon the work of Rodriguez Reggia in [17], who added both working mem-ory and a finite state machine (FSM) controller that switched the agent between different sets of movement behaviors according to its local environment and current goal. They were 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.

Attempts have been made to formalize the design of such cooperative multi-agent systems [13], [3]. In this paper, however, we turn to evolutionary computation techniques. Genetic Programming (GP) [12], Evolutionary Strategies [15], and Genetic Algorithms [8] have demonstrated the potential of computational design techniques for machine learning, optimization, and even engineering. Swarm techniques have even been applied to evolutionary computation itself as in [19], [14]. However, we are more interested in the opposite application, using evolutionary techniques to determine swarm behavior. Finite state machines have been evolved to solve problems such as automatic target detection [2]. Evolving the high level controllers therefore, seems like an extension of the design of multi-agent problem-solving that is inspired by natural design.

In this paper, we have improved the human design of particles, particularly those described in [17] by applying evolutionary techniques to

the agent control mechanisms. A formal human design technique determines the structure of the FSM and the parameters defining each state through empirical techniques and inspection. In contrast, we begin with a population of randomly-generated FSM configurations, run them through a simulated problem, then evolve the population over many generations through fitness-based selection, recombination, and mutation. Our hypothesis is that the resulting FSM configuration will perform as well as or better than a configuration tuned by a human.

2 METHODOLOGY

In order to best demonstrate that an evolutionary process can generate an agent controller that is competitive with a human designed one, we have created an experimental setup that is intentionally similar to the work done in [17]. The task is for a team of simulated agents to locate and collect resources that are spread across a large two-dimensional world with periodic boundary conditions, then return the resources to a designated home base.

The individuals of the team operate as a flock- local rules determining individual movement and goals lead to emergent collective behavior. Each individual has a limited sight range and angle, the ability to communicate with other, nearby agents and a limited working memory. Particle behavior is governed by a finite state machine that determines how each particle moves given its individual state and the environment around it.

The task is competitive, as there are two teams in the environment competing for the resources and therefore the simulation is non-deterministic. Agents can mine resources not only from designated depots, but also steal from the home base of the other team. The winner of a simulation is determined as the team that has the most resources in their home base at the end of a predetermined number of time steps.

The evolutionary process uses this simulation to pit individuals created via genetic operations against the best designed human agent. The fitness of the evolved team is a function of the resources that they have collected vs. the resources the other team collected against them. The next generation of evolution is then dependent on the fitness of teams as run in the simulation. In the parlance of evolutionary computing, the genotype is the real valued vector that represents the finite state machine of the particle and the phenotype is the emergent behavior of the team as a whole as it participates in the simulation.

In this section we will present the details of the simulation, as well as the details of the evolutionary computation mechanism that searches for the optimal, most economic particle in the simulation.

2.1 Simulation

The movement behavior of individual particles is governed by local interactions of particles within a particlar neighborhood. The particle updates its velocity at each timestep based on what it obeserves in the world around it. This leads to self-organizing, emergent behavior of the flock as a combined whole of its individuals. In particular, movement behaviors are updated with six velocity components: cohesion, alignment, separation, seeking, clearance, and avoidance. Together these components enable flocking behavior, minimize collisions, and strike a balance between the exploration of the world and the exploitation of discovered minerals.

It is important to discuss each individual velocity component in detail, as these components define the movement behaviors of the states that compose the FSM controller that is being evolved. Every component has several parameters: a radius, r and alpha, α, which determine the sight distance and angle of the neighborhood of the particular component. Each component additionaly has a weight, w, which is used to determine the contribution of the component to the final velocity of the particle.

2.1.1 Velocity Components

All particles are bound by a maximum velocity, v_{max} and for the purposes of evolution, a maximum radius, r_{max} to simulate economic constraints in the world. Other notation used is as follows: $\Delta \vec{p} = \vec{p_n} - \vec{p}$ where $\vec{p_n}$ is the average

of the neighbor positions and $\vec{p}$ is the position of the agent. Similarly $\Delta\vec{v} = \vec{v_n} - \vec{v}$ where $\vec{v_n}$ is the average of the neighbor velocities and $\vec{v}$ is the velocity of the agent. Other positions include $\vec{p_e}$ which is the position of an enemy agent and $\vec{p_t}$, the position of a target and $\vec{p}_\perp$ represents a unit vector ortogonal to the agent in the $+z$ direction. Specific velocity components are defined as follows.

Cohesion (1): a vector pointing towards the average position of friendly agents in the neighborhood, with a magnitude proportional to the agent's distance to the center of its neighbors. This component increases quadratically with distance until it is equal to the maximum velocity when the distance to the neighbor center is r. The neighborhood includes all team members who are not guarding or stunned. This velocity component ensures that flocks form and stay together throughout movement.

$$\vec{v_c} = v_{max}\frac{\Delta\vec{p}}{\|\Delta\vec{p}\|}\left(\frac{\|\Delta\vec{p}\|}{r}\right)^2 \tag{1}$$

Alignment (2): a vector in the same direction as the average velocity of friendly agents in the neighborhood, with magnitude proportional to the agent's distance to the center of its neighbors. This component increases quadratically with distance until it is equal to the maximum velocity when the distance neighbor center is r. The neighborhood includes all team members if they are seeking or spreading. Alignment helps the flock travel in the same direction in a coordinated fashion.

$$\vec{v_{al}} = v_{max}\left(\frac{\|\Delta\vec{p}\|}{r}\right)^2\frac{\Delta\vec{v}}{\|\Delta\vec{v}\|} \tag{2}$$

Separation (3): a vector pointing away from the average position of friendly agents in the neighborhood, with magnitude inversely proportional to the distance from the agent to the center of its neighbors. This component decreases quadratically with distance until it is zero when the distance to the neighbor center is r. The nighborhood includes all team members no matter their state. Separation prevents particles in the same team from colliding with each other, as well as distributing the flock outwards.

$$\vec{v_d} = -v_{max}\frac{\Delta\vec{p}}{\|\Delta\vec{p}\|}\left(\frac{r - \|\Delta\vec{p}\|}{r}\right)^2 \tag{3}$$

Seeking, Homing, and Mineral Cohesion (4): vectors pointing directly towards a target with a magnitude proportional to the maximum velocity. These velocity components allow the particle to navigate in a meaningful manner. The target is stored in memory and this velocity component isn't influenced by nearby neighbors.

$$\vec{v_h} = v_{max}\frac{\vec{p_t} - \vec{p}}{\|\vec{p_t} - \vec{p}\|} \tag{4}$$

Clearance (5): a vector pointing in a direction orthogonal to the difference between the average neighbor position and the agent's position proportional the the maximum velocity. This component ensures that swarms have a wide field of view by spreading the flock laterally along the front of movement. The neighborhood of this component is team members in sight that are not guarding or stunned.

$$\vec{v_{cl}} = v_{max}\frac{\Delta\vec{p}_\perp}{\|\Delta\vec{p}_\perp\|} \tag{5}$$

Avoidance (6): a vector pointing away from an enemy agent in the neighborhood, with magnitude v_{max} 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.

$$\vec{v_{av}} = \sum_{i=0}^{n} v_{max}\left(\frac{r - \|\Delta\vec{p_e}\|}{r}\right)\frac{\Delta\vec{p_e}}{\|\Delta\vec{p_e}\|} \tag{6}$$

2.1 Movement Behavior

The FSM controlling each agent in our first experiment contained four states: searching for new resources (SPREADING), moving to the last known resource deposit (SEEKING), carrying resources back to the home location (CARAVANING), or guarding the home base or a resource deposit (GUARDING). Each state was composed of some combination of the previously described velocity components.

To update the velocity of a particle at time t, a simple linear combination of velocity components was used. The components, along with their parameters and weights depended on the current state of the particle. The magnitude (or length) of the vector was maximized at v_{max} in the direction of the computed component velocity. Our simulation also implemented "inertia", whereby the velocity was not combined from zero, but rather added to the velocity at time $t-1$. The position of the particle at time t was simply the sum of the position at time $t-1$ with the new velocity.

$$\vec{v_t} = \vec{v_{t-1}} + \sum_i^c w_i v_i \qquad (7)$$

$$\vec{p_t} = \vec{p_{t-1}} + \vec{v_t} \qquad (8)$$

Transitions between states were triggered by conditions in each agent's local environment according to rules in the FSM (in our first experiment, these rules were fixed). 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 (which is characterized by movement toward the top location on its stack).

One of the main findings in [17] 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 outperformed 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 an enforced 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.
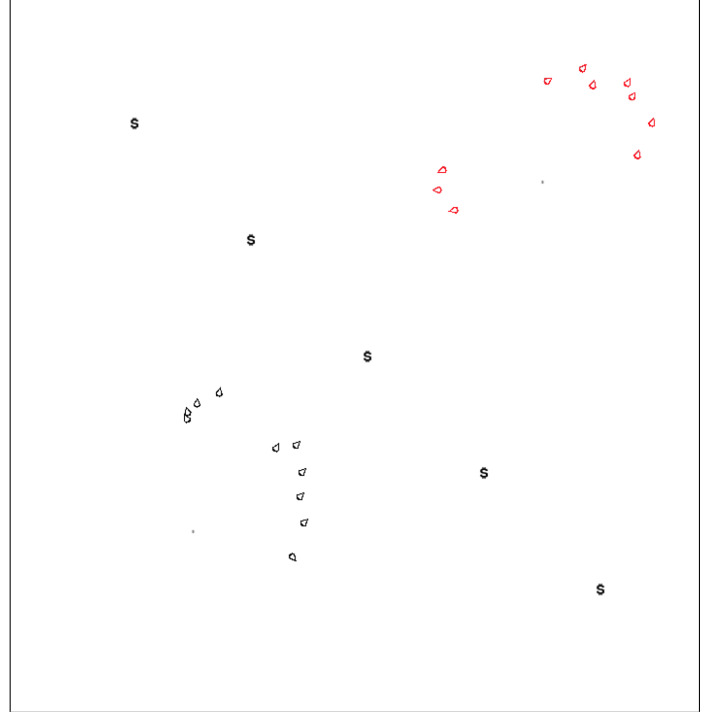

Fig. 1: Simulation of agents spreading in flocks.

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*, which 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, and we will explore adding this component in future work.

A single timestep in the world can be characterized as follows. 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 compo-

nents are linearly summed, and the resulting velocity is subjected to the v_{max} 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. All agents make this computation simultaneously, updating does not affect the world in the current time step.

2.2 Evolution

The evolution of a finite state machine for movement control was achieved through the application of a simple set of genetic operators to the states of the FSM as defined by a collection of velocity components and their behavior specific parameters. The genotype for our simulation was represented as a vector containing two integers, one for each guarding threshold, and real-valued numbers consisting of the alpha parameter, radius, and weight given to each component of each compound movement behavior.

In this experiment, the set of states in each controller was fixed to be the same set of compound behaviors as those found in [17], but with parameters derived through evolutionary programming. This allowed for easy comparison with the authoritative, human-derived solution. In addition, the transitions between states were kept the same except for the guarding thresholds, which were evolved. The basic evolutionary computation was a mixture of Evolutionary Strategy and Evolutionary Programming techniques and is characterized in Table 1.

The mutation operator first selected which parameters to modify independently. We used a mutation probability parameter $p_m = 0.2$ for mutating each weight, radius, alpha, and threshold. Each selected threshold had an equal chance of increasing by 1 and decreasing by 1. Each selected real-valued parameter was modified through the addition of a uniformly distributed random number with mean 0. We mutated each weight with world constraints using $r \in (1, 500]$ and $\alpha \in (0, 360]$, the step size of the mutation was $m_w \in [-0.2, 0.2]$ for

Genotype	Real valued vectors representing FSM states
Phenotype	Swarm behavior from individual movements
Selection	Tournament with Elitism
Recombination	Intermediary exempting elites
Mutation	Linear Rule exempting elites
Fitness	Resources collected after 10k timesteps
Specialty	Adpating FSM with same size network

TABLE 1: Characterization of Evolutionary Method

weights and $m_s \in [20, 20]$ for the r and α parameters. Elite individuals were exempted from mutation.

The recombination operator also ignored the elite individuals carried over from the last population. Parents were selected with a recombination probability, $p_r = 0.2$, to be recombined with another individual. We used intermediary recombination of the weight, radius, and alpha parameters of all components in all states, bounded by the minimum and maximum possible values of each parameter. The second chosen parent remained in the population. Although elite individuals were exempt from recombination, they were able to be selected as the secondary parent, contributing their genetic material to the recombined offspring.

A round of evolution consisted of the evaluation of every individual followed by the application of elitism, tournament selection, recombination, and mutation to produce the next generation. For elitism, the N_e individuals of the previous generation was always carried into the next generation without modification. The other $p - N_e$ individuals were selected via tournaments of size 3. All non-elite individuals chosen through the tournament then each had a p_r chance of undergoing recombination with another randomly chosen child. Each of the resulting non-elite individuals was then subjected to the mutation operator, mutating parameters independently.

3 EXPERIMENTAL PROCEDURE

Our computational experiment required two main components; a system to carry out the evolutionary computation, which constructed genotype populations via genetic operators, and a simulation system to instantiate

phenotypes- particle swarms whose individuals are constructed from the genotypes. The simulation system computed the fitness of a particular swarm, which allowed the evolutionary computation to build the next generation.

The simulation environment consisted of two simulated teams in competition with each other; a red team and a black team. The simulated world was a 3000 x 3000 world with periodic boundary conditions and the number of timesteps was limited to 10,000 per simulation due to computational constraints. We restricted each team to 10 agents each,and evenly distributed 5 resource deposits betwen the red and black bases with 80 resources per deposit. Both the team size and the time limit were implemented to make the evolutionary computation tractable; our implementation required on average 273 seconds to complete; and each generation took approximately 52 minutes to compute all fitness values.

The red team's behavior was implemented using a finite state machine, memory, and movement parameters exactly as the best empirical result designed by humans in [17], called *home guarding flock* meaning that the swarm guarded both its home base but not discovered mineral deposits. The black team utilized the configuration derived from a genotype individual produced by the evolutionary computation. We attempted to hold as many variables constant as possible, including the locations of bases and resources, starting positions and velocities of agents, and update order. After 10,000 timesteps, the fitness of the black team was returned as the number of resources in the black base.

Each simulation was run in parallel as subprocess of a master evolutionary computation process. The master process randomly initialized a population of 50 black team configurations, then ran a simulation for each individual in the population. After all 50 simulations were completed, tournament based selection with tournaments of size three were carried over to the next generation along with the 5 most elite individuals. After selection, genetic operators were applied.

In our first experiment we only used mutation to change the states in the finite state machine of the particles. In the second experiment we used recombination in addition to mutation operations. In either case, elite individuals were exempt from genetic operations, however elite individuals could be recombined to replace other individuals in the population. Both the mutation and recombination (when used) had a likelihood of 0.2. Maximum and minimum parameters were also placed on the values of the weight, the radii, and the alpha values to ensure an economic and reasonable result.

Simulation	Mean	Std Dev	Max	Min
human	119.0	43.2	274	119
evolved	296.0	25.6	336	254
random	0.3	0.67	2	0

TABLE 2: Aggregated fitness of various particles competing with human designed particles.

4 RESULTS

In this section we will evaluate the effectiveness of both evolved flocking particles vs. human created particles, as well as the ability of an evolutionary computation to design such particles. In short, the evolutionary computation mechanism was able to design a particle system that exhibited variations in behaviors that were different enough from the human designed particles so as not only be competitive with the human design team, but able to defeat the other team in terms of resources collected.

Table 2 shows the mean, standard deviation, maximum, and minimum resources collected for the black team over ten runs of the simulation with the same parameters. Black teams of comparsion are the human designed team, the best evolved team, and a random team. The evolution designed team performs significantly better than the human designed team.

The average fitness over time of the black and red teams is displayed in Figure 2 which graphs the mean fitness of ten individual runs of the best human designed team (the red team) vs the best evolved team, the black team. Within 6000 timesteps the red team begins to lose resources as the black team exploits the red base in addition to more resource locations
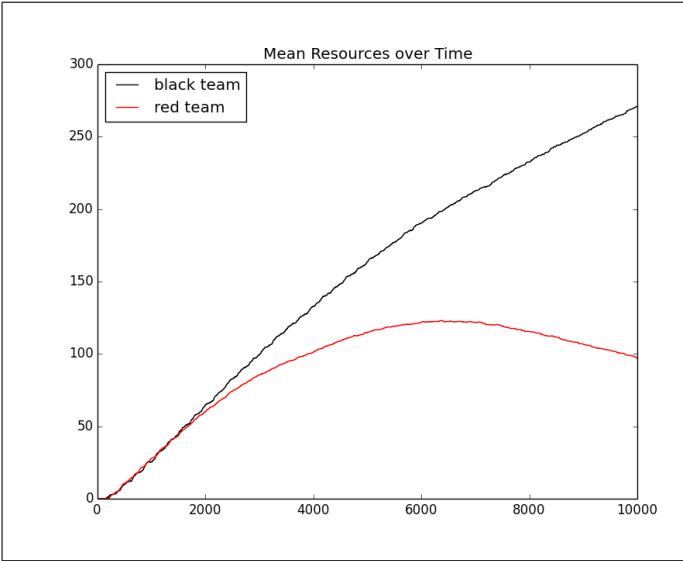
Fig. 2: Performance of evolved particles vs human designed particles in head to head simulation of 10,000 timesteps

Fig. 3: Mean fitness per generation with mutation

Fig. 4: Mean fitness per generation with mutation and recombination

than the red team, which favors a strategy of exploiting one resource deposit at a time.

The performance of our two evolutionary computations is expressed in Figures 3 and 4 which plots the mean fitness of the entire population per each generation. In experiment one, only the mutation operator was used to generate the next population. In experiment two, both recombination and mutation were used to produce the next generation.

As is shown from the graphs; the first experiment was wildly unstable in terms of its search across the optimal state space. This may have been due to the fact that highly variable mutation caused dramatic shifts in the behavior of the phenotypes. It also may have been due to a programming error on the part of the research team. In the second evolution with recombination as well as mutation, there is more steadily growing fitness over time and a more stable population to select from to apply genetic operations.

5 DISCUSSION

The evolution designed particles surpassed their human design counterparts through a qualitatively novel mechanism; the human designed team prefered to stay in one or two larger flocks; whereas the high performing evolved teams split into more, smaller flocks
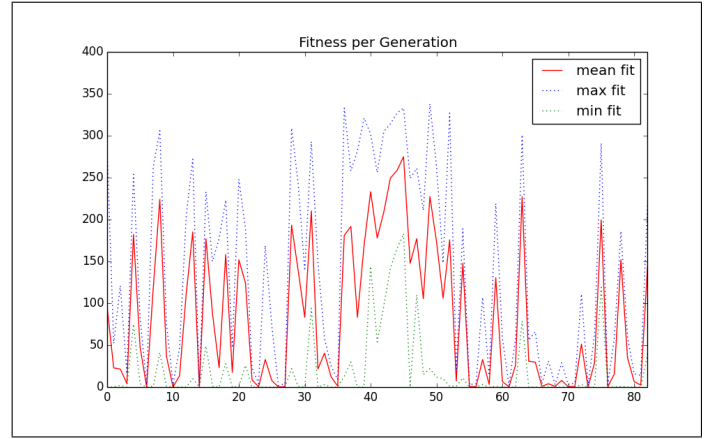
and were able to exploit the search space more effectively.

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] Karl Benson. Evolving finite state machines with embedded genetic programming for automatic target detection. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 2, pages 1543–1549. IEEE, 2000.

[3] Davy Capera, JeanYPierre George, 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.

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

[5] 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.

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

[7] 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.

[8] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

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

[10] 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.

[11] 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.

[12] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[13] 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.

[14] 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.

[15] Ingo Rechenberg. Evolution strategy: Natures way of optimization. In *Optimization: Methods and applications, possibilities and limitations*, pages 106–126. Springer, 1989.

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

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

[18] 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.

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

[20] 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.

[21] 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.