

Adaptive Problem Solving with Particle Systems

Alejandro Rodríguez and James A. Reggia

Dept. of Computer Science and UMIACS

University of Maryland

College Park, MD 20740

{alejandr,reggia}@cs.umd.edu

ABSTRACT

Self-organizing particle systems (“swarms”) consist of numerous autonomous, reflexive agents (“particles”) whose collective movements through space are determined primarily by local influences. We are currently extending particle systems so that they not only move collectively, but also solve simple problems. This is done by giving the individual particles/agents a rudimentary intelligence in the form of a limited memory and a top-down, goal-directed control mechanism. Such enhanced particle systems are shown to be able to function effectively in performing simulated search-and-collect tasks. However, measuring the effects on particle collective performance of different design choices for individual agents proved to be difficult. We resolved this issue by allowing different agent teams to compete with one another under a variety of controlled conditions. This allowed us to demonstrate clearly how different agent features (independent vs. coordinated movement, exploratory vs. protective behaviors) impacted the behavior of the collective as a whole.

1 INTRODUCTION

Recent work creating computational models of coordinated movements by collections of locally interacting individuals includes, for example, models of bird flocks [12, 13], fish schools [6, 16], social insect swarms [2], and self-assembling molecules [4]. All of these systems consist of numerous autonomous “particles” (birds, ants, vehicles, etc.) whose movements through space are governed by primarily local “forces” exerted on them by other nearby particles or the environment. Methodologically-related approaches such as particle swarms [8], cultural algorithms [3], and bacterial chemotaxis algorithms [10] have generalized this idea to abstract, n-dimensional “cognitive spaces”. Here we will refer to all of these past models as *self-organizing particle systems*.

Typically, these systems consist of mainly reactive particles whose behavior is completely determined by reflexive movement dynamics. Interactions between these particles result in remarkably complex global behavior which

emerges from the joint actions and relatively simple behaviors of the individual particles, thus exhibiting self-organization. These properties have led to applications in computer graphics [12, 13], multi-robot team control [1, 5, 11, 17, 18], and numerical optimization [8]. The simplicity of the particles makes their use difficult for tasks other than the simulation of movement patterns. However it has been proposed in a few studies that it is possible to extend the paradigm towards autonomous problem solving, for example using multi-robot teams for pushing objects [9] or foraging [7] or more general tasks as the Robocup competitions [17].

The ultimate goal of our research is to extend particle systems to act as more general problem-solving entities that not only move in a coordinated fashion, but also collectively set goals and make decisions, thereby forming a *self-organizing collective intelligence* [15, 19]. By this term we mean a self-organizing particle system where the individual particles are no longer mindless, but have at least a limited goal-driven intelligence and decision making ability. To achieve this, the low-level movement dynamics used in past particle systems is complemented by a top-down, goal-driven control mechanism capable of simple inferences and of autonomously switching between different formulations of its movement dynamics. Since such enhanced particles have simple inference abilities that influence their movements, we will also refer to them as *agents* that form a *team*. In this paper, we describe our first steps in this work: combining a finite state machine (FSM) control process that selects goals and switches between alternate movement dynamics as the agent collective solves “search and collect” problems.

Given the self-organizing and interactive nature of such systems, measuring the impact of design choices for individual agents on the performance of the system as a whole poses a particularly difficult problem since the contribution of the individual components to global behavior cannot be easily determined. We successfully addressed this problem by systematically varying specific agent features and comparing the results through a series of competitions between different agent teams.

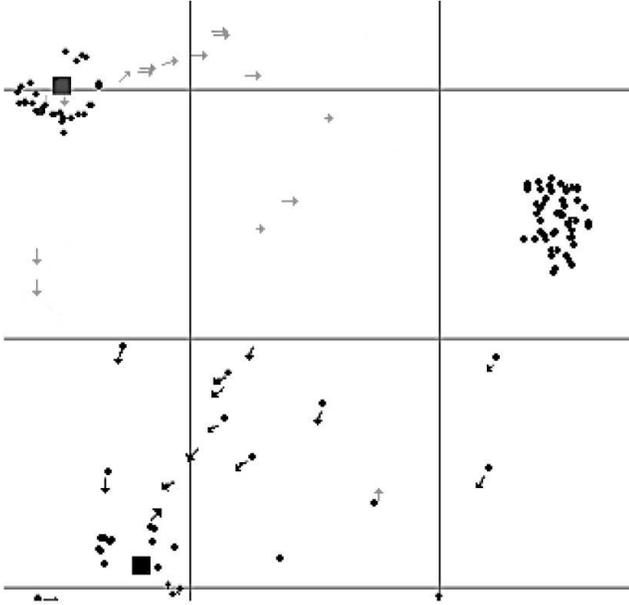


Figure 1: A small section of a 3000 x 3000 continuous world with a single “mineral deposit” on the right and two different teams (dark and light arrows) exploiting it. Teams’ homes are denoted by small solid squares and mineral units as spots. The dark team agents are returning home carrying minerals (spots adjacent to arrows). Most agents of the light team are returning to the deposit after unloading minerals at home, but some of them are simply exploring.

2 METHODS

As a first test of the benefits of collective rather than individual problem-solving, we used a resource collection task. A team of homogeneous agents is initially deployed at a random location inside a 2D world with periodic boundary conditions (see Figure 1). In this world, units of some resource that we call minerals (but which could also represent any other valuable material) are located randomly. A few areas are heavily dense in mineral, thus serving as “deposits”, while the rest of the minerals are scattered throughout the world. The task of the agents is to collect the minerals and deposit them in a designated “home” location. The success of a team is measured in terms of the amount of minerals accumulated at its home over time, which advances in small discrete steps. Other teams may be present in the world, and members of one team can potentially hinder rival teams by different means, such as blocking them (agents are not allowed to collide with others), or even stealing from their home.

Agents have a two layer architecture. The bottom level layer (local information plus movement dynamics) controls the reactive behavior of the agent, making instantaneous

decisions about the actions to be performed. It takes input solely from the local environment at a given instant, and outputs a corresponding action based on the immediate goals of the agent and current (local) state of the environment; this is similar to what has been done in past particle systems. The top layer, not found in past particle systems, consists of a very limited memory and a finite state machine (FSM) that directs agent behavior in a top-down fashion, modifying the movement dynamics used over time. For example, if the FSM decides that it is time for the agent to go home, it will switch to the state *carrying* and provide the bottom layer with the target location of its home location. The bottom layer will then determine at each step the steering actions needed to properly navigate from the current location to the home. Since the bottom layer is mostly reactive it can temporarily override the long term goal of going home for a more pressing need, such as avoiding a competing agent or obstacle.

Other agents and objects in the world are visible to each agent as long as they are within its local neighborhood. The neighborhood of an agent is defined as a circular segment of radius r and angle α in front of the agent. An agent senses the relative position, orientation and current velocity of every object in its neighborhood, in addition to its own current position. Although agents from the same team are homogeneous and interchangeable, they are able to distinguish between members of their own team and other agents. Agents may also know the state of members of their own team in their neighborhood. Agents are able to pick up, carry and put down a single resource at a time.

At any time, each agent is in one of several possible mutually exclusive states. For example, an agent could be *searching for minerals* or *carrying minerals home*. These states and the transitions between them are represented with a finite state machine (FSM). The agent’s current state determines which set of parameters for the low-level reactive controller currently drives its movements and behavior. Figure 2 shows the FSM employed in the experiments reported here and the corresponding movement behaviors associated with each state. Agents are initially in an idle state. Once they receive a “start signal”, they begin *searching* for resources. When they find some, they choose between *picking up* the minerals or *guarding* the deposit from other teams, depending on whether a group of guarders is already formed. When an agent detects five or more agents guarding in its neighborhood, it decides the deposit is already guarded. Agents recognize a deposit when they detect a certain amount of minerals in their local neighborhood. This implies that homes of other teams are interpreted as deposits given that they have accumulated enough minerals. If the agent succeeds in picking up a unit of mineral, then it starts *carrying* the minerals home. After arriving home with a unit of mineral and depositing it, if the home is unprotected agents go to *guard-*

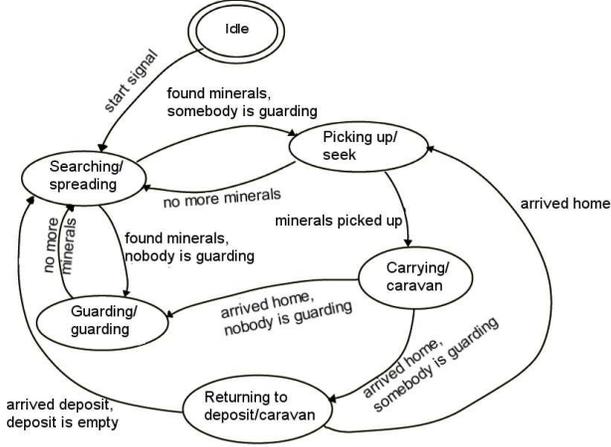


Figure 2: FSM of an agent showing its states and the movement behaviors associated with each state. States are represented by circles labeled by <State/associated controller>, while arrows represent transitions labeled by the triggering event. The initial state is marked by a double circle.

ing to protect it, otherwise they return to the last deposit they were exploiting (*returning to deposit*). If the flock arrives at its (unprotected) home, the first five agents will become guarders and the rest will return to the deposit. When an agent is guarding, it will remain so unless all of the mineral in its neighborhood is taken away (the deposit is exhausted), in which case the agent returns to *searching*. When agents are *returning to deposit*, it could happen that the deposit is already exhausted, in which case the agent goes into *searching* for another deposit, otherwise it tries *picking up* more minerals.

Agents have a very simple memory that allows them to recall the location of the deposits that the agent has found and the location of the home of the agent. The precise current goal of the agent is thus represented as a combination of the current state and the contents of this memory. For example, the state could determine that the agent is going back to a deposit to exploit it, while the memory determines which deposit the agent is to exploit.

The low level movement dynamics that guide the agent through the environment are inspired by earlier work [12, 13]. Movements are governed by a set of individual influences (avoiding an obstacle or competing agent, staying with the flock, keeping the same distance from other agents from the team, etc.) that produce an instantaneous acceleration determined by a desired velocity vector. The individual influences are combined in a non-linear fashion. By changing which individual influences are combined and their relative weights (done by the FSM), a large variety of movement behaviors can be implemented, each one associated with a different state or goal.

Seven individual influences act on each agent and are computed based on the position and velocity of neighboring agents. For instance, a *cohesion velocity* \vec{v}_c tends to move the agent toward the center of the flock and is computed as

$$\vec{v}_c = \left(\frac{\|\vec{p}_n - \vec{p}\|}{r} \right)^2 v_{max} \frac{\vec{p}_n - \vec{p}}{\|\vec{p}_n - \vec{p}\|}$$

where \vec{p} is the current position of the agent, \vec{p}_n is the average position of its neighbors and v_{max} the maximum speed of the agent. The direction of this velocity is directly toward the center \vec{p}_n of the agents in its neighborhood, while its magnitude is a fraction of the maximum velocity that increases quadratically with the distance from that center. The remaining six velocities are explained in more detail in [15]. An *alignment velocity* tends to move the agent in the same direction that its neighbors are moving. An *avoidance velocity* tends to move the agent away from other agents. A *separation velocity* tends to move the agent away from neighbors by steering away from the center of the flock. A *seeking velocity* influences an agent to move toward an observed unit of mineral. A *clearance velocity* influences an agent to steer toward the side when there is an agent in front of it and tends to align a group of agents side by side. Finally, a *homing velocity* drives the agent to a point in the space such as the home of the agent or a remembered deposit. The individual velocity influences are combined as a weighted sum at each time step to update each agent's resulting velocity \vec{v} . The summation process is prioritized, with the terms being added in a fixed order, and when the sum exceeds a certain threshold v_{max} the term is dropped. Different prioritizations and weight values are used to produce each specific behavior of the agent. Once the new velocity of the agent has been computed, its new position is updated.

By prioritizing and weighting each of the seven velocity components differently, four different *movement behaviors* are implemented. Full details are given in [15]. Briefly, *spreading agents* tend to form a flock slightly similar to the broad, V-shaped formations that migratory birds adopt (Figure 3a). This formation is ideal for searching for deposits, since upon seeing a deposit, an agent will tend to "pull" the rest of the flock to it via local interactions (Figure 4). *Seeking agents* go after a target, for example a unit of mineral, while avoiding obstacles and other agents (Figure 3b). *Caravaning agents* move in a pattern similar to spreading agents, but they are homing on a particular point in space (e.g., a previously visited deposit) and thus have a narrower front (Figure 3c). Finally, *guarding agents* patrol a deposit (Figure 3d), distributing themselves about evenly through the deposit. The interaction between nearby guarding agents and even non-guarding agents make them keep moving, effectively orbiting the

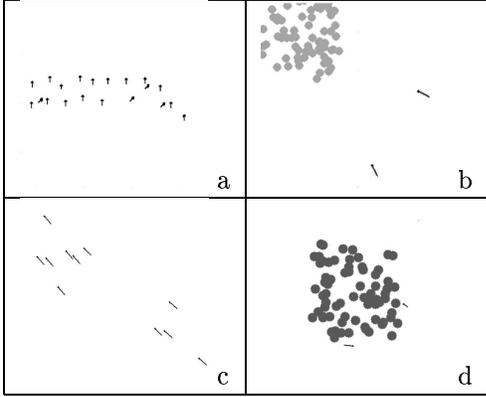


Figure 3: Collective movements. (a) Spreading agents (arrows), are located roughly side by side. (b) Two seeking agents head to the closest mineral unit (gray spots). (c) Caravanning agents tending to align. (d) Two guarding agents patrolling a deposit.

deposit. Since agents cannot bump into each other, this helps to keep intruders away by repelling them from the deposit.

Each computational experiment below consists of letting one or more simulated teams search for the minerals in the world for a limited time. To compare the effectiveness of collective movement behaviors versus independently moving agents, a set of independent agents have been implemented. These agents follow the same model explained above, with the important exception that they do not take into account other agents in their dynamics. The controllers for these agents replace dependent steering behaviors (cohesion, alignment, separation, clearance) with random wandering. Collision is not replaced. Otherwise the FSM of the non-flocking (independent) agents is the same as for the collectively moving agents (Figure 2).

The results reported below are the average over 20 runs for each experiment, using 50 agents per team (unless indicated otherwise) in a continuous world of size 3,000 x 3,000 and one team for every type of agent (described below). During each run, 12 deposits were randomly located and independently created, each deposit consisting of 80 units of minerals. These parameters were chosen so that it is valuable for a team to go back to a deposit repeatedly. Also, the number of deposits and size of the world make it easy for agents to find the deposits (and other agents' home), but still force agents to compete for the resource (minerals). Teams' homes were also independently located at random in each run and agents were initially randomly positioned. In each run, the simulation lasted 40,000 time-steps (iterations), which was adequate for the average amount of mineral in teams' home to converge to some value.

Multiple teams of agents were often used in an exper-

imental run, with each team being of a different type. Which teams were involved could vary from experiment to experiment. The six different types are:

- Full-guarding flock. Collectively moving agents which guard home and any deposits that they find.
- Home-guarding flock. These agents will not guard a deposit, but will still guard their own home.
- Non-guarding flock. These agents are the same as above except they do not have a guarding state.
- Full-guarding, home-guarding and non-guarding independent teams. These three types of agents correspond respectively to the three types of flocking agents above, but do not undertake collective movements; they move independently. They search through random wandering and see other agents only as obstacles to be avoided.

3 RESULTS

In the first experiment, a team is present in the world without any other competing teams. The experiment was repeated for each team. Figure 5 shows the amount of minerals collected over time, averaged over 20 runs. After 20,000 iterations most teams have succeeded in collecting almost all of the minerals available in the world, with the exception of the full-guarding teams that are still slowly collecting resources. This is due to the fact that after the full-guarding teams split off members to guard every deposit found, not enough agents remain to collect the minerals rapidly. The non-guarding flocking team seems to be the fastest in collecting minerals, which is consistent with the fact that all team members are actively searching for and collecting minerals, and that in the absence of other teams, guarding mineral deposits has no value. In this world where agent teams do not need to compete with other teams, there is no clear difference between the other four teams.

In the second experiment, all six types of agents (one team per type) are present simultaneously in the same simulation. The number of agents per team was decreased to 30 to avoid overcrowding. Figure 6 shows the amount of minerals in each teams' home over time, averaged over 20 runs. Most striking is that the home-guarding, flocking team's collected resources increase monotonically, with this team clearly outperforming all others. Early in simulations (during the first 5,000 iterations), both this and the non-guarding flocking team collect minerals faster than any other team. After the first few thousand iterations, the explored area of each team is wide enough for teams to find each others' homes. Accordingly, the amount of minerals

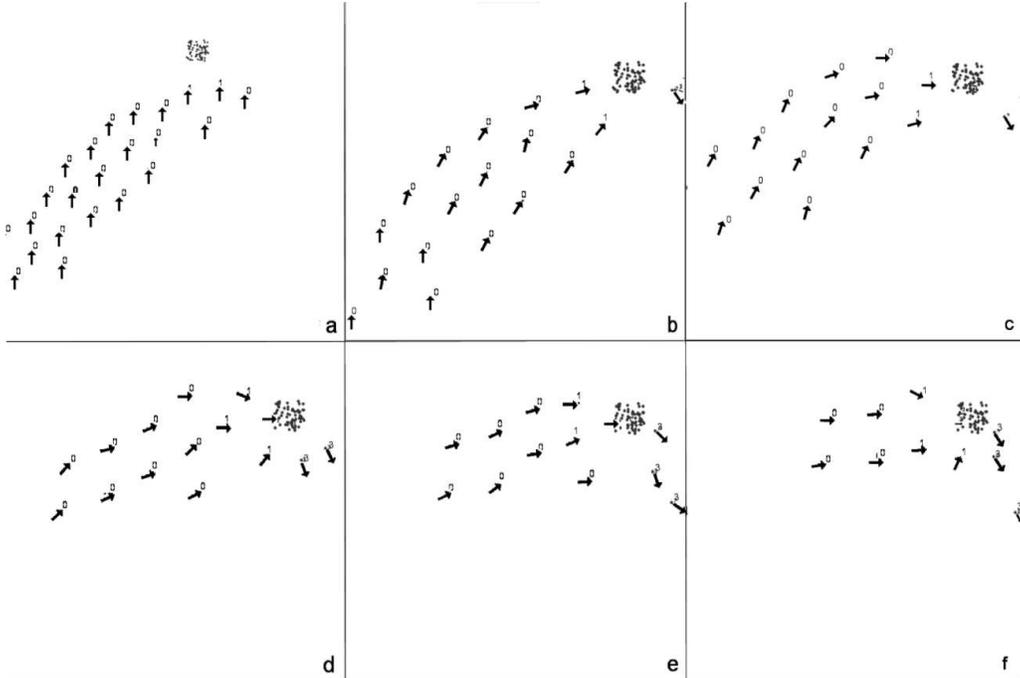


Figure 4: Agents in a flock being pulled toward a deposit. The number on top of each agent represents its current state (0 for Searching for a deposit, 1 for Picking up). Only agents in state 1 actually detect the deposit. At *a*, only two agents have located the deposit, while the rest of the flock moves northward. At *b* and *c*, agents that are near the deposit but that do not yet see it turn toward those agents that have seen the deposit and are already going toward it. From *d* to *f*, the whole flock gradually turns toward the deposit and collects minerals. Such behavior indicates an advantage of collective movements in recruiting other agents to carrying a resource when it is discovered by just a few.

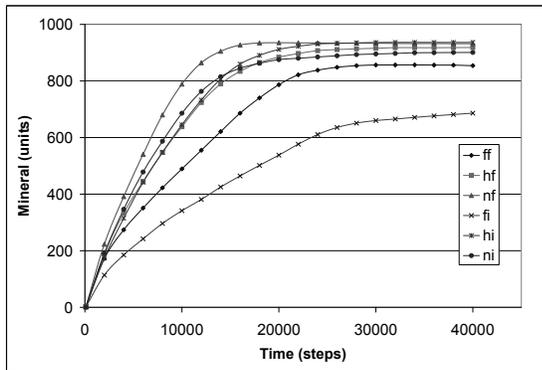


Figure 5: Mineral collected over time by each team alone. ff: Full-guarding flocking, hf: home-guarding flocking, nf: non-guarding flocking, fi:full-guarding independent, hi:home-guarding ind., ni: non-guarding ind.

decreases in subsequent iterations for most teams, especially the non-flocking teams. The differences in the mean amount of collected minerals by each team after 40,000 iterations over 20 runs are statistically significant at the level of 95% according to a two-way ANOVA, both in sociality (flocking vs independent) and guarding strategy (full-guarding, home-only and none). These data suggest two main hypotheses. First, teams of collectively moving agents are more effective at this task than corresponding teams of independently moving agents. With collectively moving agents, whenever a deposit was discovered by an agent, numerous other agents were immediately nearby and thus pulled in by local inter-agent influences to help collect the discovered minerals (e.g., see Figure 4). Second, for both collectively and independently moving agent teams, agents that guarded only their home did better than non-guarding agents, who in turn did better than full-guarding agents. Presumably allocating agents to guard resources, especially multiple deposits, has a large cost: it removes these agents from collecting minerals, and this loss is not adequately compensated for by any protective influences they exert through their blocking actions.

The impact of collective versus independent movements on agent teams can be clarified by varying just that factor

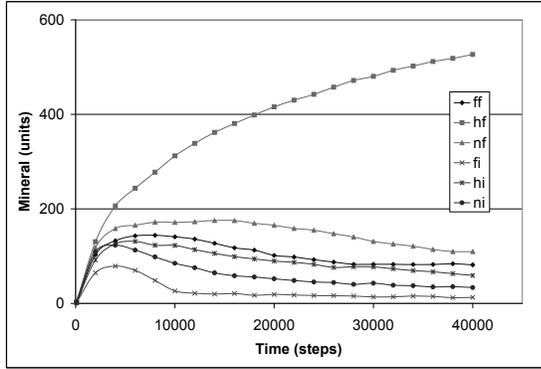


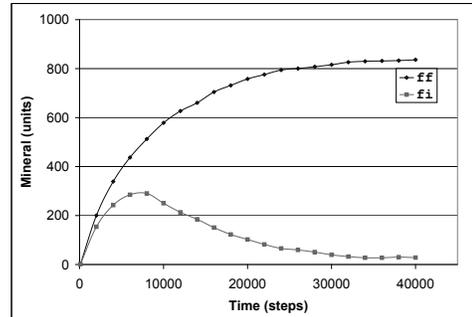
Figure 6: Mineral present in teams' home per unit of time when all teams compete simultaneously in a single world.

between two competing teams. Figure 7 shows the mean amount of mineral saved at home over time for pairwise competitions of collectively moving versus independently moving teams of agents. These results are significant at the level of 95%. It is clear that the flocking teams are always faster in accumulating minerals. Even more striking, the independently moving teams are not sufficiently effective in protecting their home from being looted by the flocking team, and their collected minerals considerably decrease during the following iterations.

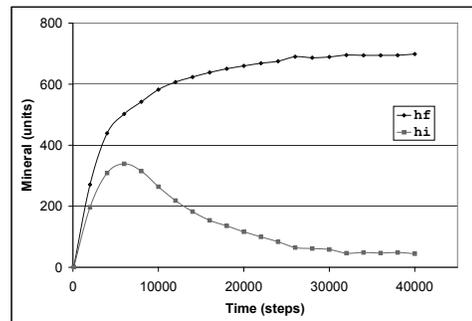
Finally, experiments to compare the matched pairs of guarding versus non-guarding teams were performed. Figure 8 shows the mean amount of mineral saved at each team's home over time. Results are significant at the level of 95%. Early on in the simulation (about iteration 5,000) pairwise teams have similar performance, but after this guarding teams show a clear advantage, as their amount of minerals saved at home keep increasing, while the amount of minerals in the home of non-guarding teams decreases, probably as it is taken by the opposite team. Again, home-guarding teams perform better than full-guarding teams.

4 DISCUSSION

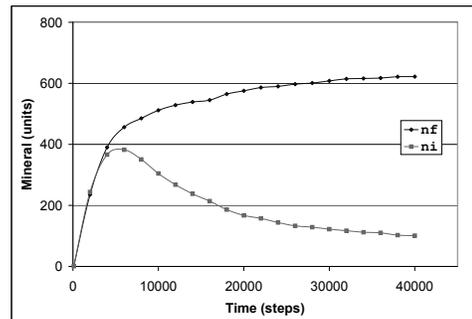
In this paper, we have examined the question of whether self-organizing particle systems can be extended to exhibit behaviors more general than just collective movements. Specifically, our hypothesis was that by giving the normally purely reflexive agents found in particle systems a few behavioral states, a simple finite state transition graph that governs state changes, and a simple memory of the locations of significant objects that are encountered, the resulting agent team would have the ability to collectively solve resource locate-and-collect problems. Individual behaviors are implemented by letting each state of an agent be associated with both a different goal and with a corresponding set of parameters that influence the agent's



(a)



(b)



(c)

Figure 7: Mean mineral collected over time by full-guarding teams. (a) ff: full-guarding flocking versus fi: full-guarding independent agents. (b) hf: home-guarding flocking versus hi: home-guarding independent agents. (c) nf: non-guarding flocking versus ni: non-guarding independent agents.

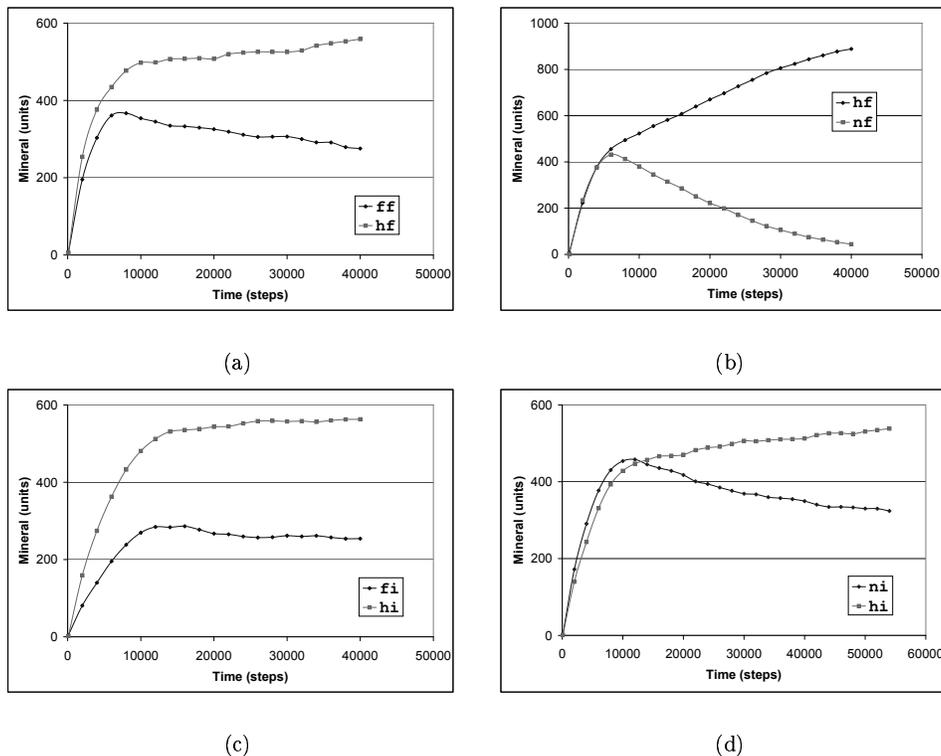


Figure 8: Mineral present in teams' homes per unit of time. (a) ff: full-guarding flocking versus hf:home-guarding flocking. (b) hf: home-guarding flocking versus nf:non-guarding flocking. (c) fi: full-guarding independent versus hi:home-guarding independent. (d) hi: home-guarding independent versus ni:non-guarding independent.

movements. This effectively couples the collectives' goals to different movement dynamics. Under such conditions, where state changes are triggered by environmental events and the states of other nearby agents in a way that retains the local nature of information processing in particle systems, one would anticipate the emergence of problem-solving abilities by an agent team as a whole.

The simulation results presented in this paper and our earlier work [15, 19] provide substantial support for our hypothesis. As state changes occurred and spread throughout a collection of agents via local interactions, the group's motion as a whole was influenced and shifted to provide collective problem-solving. An agent team could routinely search for, collect, and return discovered resources to a predetermined home location, all the while retaining movement as a "flock" of individuals. Further, it was found in simulations that a team of agents that moved collectively was more effective in solving search and collect problems than very similar agents that moved independently. This was because when one or a few agents on a collectively-moving team discovered a site with plentiful resources, they would automatically pull other team members toward that site, greatly expediting the acquisition of the discov-

ered resource. Thus, a benefit of underlying collective movements of particle systems which, to our knowledge, has not been appreciated in past work, is that they have the potential to automatically recruit additional agents to complete a task when one agent detects the necessity of that task.

We also undertook computational experiments in which multiple teams with somewhat different behaviors simultaneously competed to find and collect the resources that were present. Regardless of whether the teams competed two at a time or all at once, we found consistently that collectively moving agents were superior to independently moving teams of matched agents in collecting resources. Further, and regardless of whether agents moved as a team or independently, we found that those that were allowed to guard only their home base did best, those that tried to guard both home and discovered resources did worst, and those that guarded nothing were in between. This finding reflects a kind of exploration/exploitation trade off: guarding has a protective value for preserving located and collected resources, but also a cost in that fewer agents are available to continue searching for and collecting new resources. Most importantly, these simulations exhibited

group-level decisions not just about which type of movements to make, but also about when it was appropriate to split into groups, with one smaller group remaining to guard resources. They exhibited decentralized cooperation without explicit coordination, such as when a wandering agent would follow another agent that knew the location of uncollected resources, simply because wandering agents tended to follow other agents. Our results, as well as related ongoing work [19], show that the reflexive agents of contemporary particle systems can readily be extended to successfully support goal-directed problem solving while still retaining their collective movement behaviors.

Acknowledgments: This work was supported by NSF ITR award ITS-0325089.

REFERENCES

- [1] Balch, T., & Arkin, R. (1998). Behavior-based Formation Control for Multi-robot Teams. *IEEE Robotics and Automation*, 14(6), 926-939.
- [2] Bonabeau, E., Dorigo, M., & Theraulz, G. (1999). *Swarm Intelligence*. Oxford Univ. Press.
- [3] Chung C. & Reynolds R. (1996) A Testbed for Solving Optimization Problems Using Cultural Algorithms, *Evolutionary Programming*, 225-236.
- [4] Edwards, L., Peng, Y., & Reggia, J. (1998). Computational Models for the Formation of Protocell Structures. *Artificial Life*, 4, 61-77.
- [5] Fredslund, J. & Mataric M. (2002). A General Algorithm for Robot Formations Using Local Sensing and Minimal Communication. *IEEE Transactions on Robotics and Automation*. 18(5), 837-846.
- [6] Huth, A., & Wiesel, C. (1992). Simulation of Movement of Fish Schools. *J. Theoret. Biol.*, 156, 365-385.
- [7] Jones, C. & Mataric, M. (2003). Adaptive Division of Labor in Large-Scale Minimalist Multi-Robot Systems. *Proc. IEEE International Conference on Intelligent Robots and Systems*, 1969-1974.
- [8] Kennedy, J., & Eberhard, R. (2001). *Swarm Intelligence*. Academic.
- [9] Kube, C., & Zhang, H. (1992). Collective Robotic Intelligence, *Second International Conference on Simulation of Adaptive Behavior*, 460-468.
- [10] Muller, S., Marchetto, J., Airaghi, S., & Kourmoutsakos, P. (2002). Optimization Based on Bacterial Chemotaxis. *IEEE Transactions on Evolutionary Computation*, 6, 16-29.
- [11] Parker, L. (1993). Designing Control Laws for Cooperative Agent Teams. *Proc. IEEE International Conference on Robotics and Automation*, 3, 582-587.
- [12] Reynolds, C. (1987). Flocks, Herds and Schools. *Computer Graphics*, 21, 25-34.
- [13] Reynolds, C. (1999). Steering Behaviors for Autonomous Characters. *Proc. Game Developers Conf.*, 763-782.
- [14] Reynolds, C. (2000). Interactions with Groups of Autonomous Characters. *Proc. Game Developers Conf.*, San Francisco: CMP Game Media Group, 449-466.
- [15] Rodríguez A. & Reggia J. (2004) Extending Self-Organizing Particle Systems to Problem Solving, *Artificial Life*, 10(4), 379-395.
- [16] Tu, X. & Terzopoulos, D. (1994). Artificial Fishes: Physics, Locomotion, Perception, Behavior. *Proc. ACM SIGGRAPH*, 42-48.
- [17] Vail, D., & Veloso, M. (2003). Multi-Robot Dynamic Role Assignment and Coordination Through Shared Potential Fields. In *Multi-Robot Systems*. Schultz, A., Parker, L., & Schneider, F. (editors). Kluwer.
- [18] Verth, V., Brueggemann, V., Owen J., & McMurry P. (2000). Formation-Based Pathfinding with Real-World Vehicles. *Proc. Game Developers Conference*.
- [19] Winder R. & Reggia J. (2004) Distributed partial Memories to Improve Self-Organizing Collective Movements, *IEEE Trans. SMC B*, 34, 1967-1707.