Improving Swarm Survival Using DAEDALUS

Suranga Hettiarachchi
Computer Science Department
Indiana University Southeast
New Albany, IN 47150, USA

Abstract

Traditional approaches to designing multi-agent systems are offline (in simulation), and assume the presence of a global observer. In the online (real world), there may be no global observer, performance feedback may be delayed or perturbed by noise, agents may only interact with their local neighbors, and only a subset of agents may experience any form of performance feedback. Under these circumstances, it is much more difficult to design multi-agent systems. DAEDALUS is a framework designed to address these issues, by mimicking more closely the actual dynamics of populations of agents moving and interacting in a task environment. The agents modeled with DAEDALUS use mutation as a mechanism to improve their performance in the task environment. This paper explores 1) how agent-agent interaction such as receiving rules and mutation rate from another agent during navigation and 2) receiving mean mutation rate computed using agent's better performing neighbors' mutation rates when agents are at a goal affect swarm survival (i.e. agents reaching a goal within a predefined time interval), and presents a detail analysis showing how DAEDALUS can be used to improve swarm survival in an online environment. In our task environment, a swarm of agents navigate through an obstacle field towards a goal while maintaining a formation, where the obstacles can obstruct their perception.

1. Introduction

Engineering multi-agent systems is difficult due to numerous constraints, such as noise, limited range of interaction with other agents, delayed feedback, and the distributed autonomy of the agents. One potential solution is to automate the design of multi-agent systems in simulation, using evolutionary algorithms (EAs) (Grefenstette 1989; Wu *et al*. 1999). In this paradigm, the EA evolves the behaviors of the agents (and their local interactions), such that the global task behavior emerges. A global observer monitors the collective and provides a measure of performance to the individual agents. Agent behaviors that lead to desirable global behavior are hence rewarded, and the collective system is gradually evolved to provide optimal global performance.

There are several difficulties with above evolutionary approach. First, a global observer may not exist. Second, some (but not all) agents may experience some form of reward for achieving task behavior, while others do not. Third, this reward may be delayed, or may be noisy. Fourth, the above paradigm works well in simulation (offline) but is not feasible for real-world online applications where unexpected events occur. Finally, the above paradigm may have difficulty evolving different individual behaviors for different agents (heterogeneity vs homogeneity).

In our prior work (Hettiarachchi *et al*. 2006), we introduced "Distributed Agent Evolution with Dynamic Adaptation to Local Unexpected Scenarios" (DAEDALUS), for engineering multi-agent systems that can be used either offline or online and showed how DAEDALUS can be used to achieve global aggregate behavior of agents that move through an obstacle field towards a goal. The obstacles obstruct the perception of the agents (i.e. they act to degrade the interactions between agents).

In (Hettiarachchi and Spears 2006), we presented a comparative analysis of homogeneous and heterogeneous swarm learning with and without mutation exchange. To overcome navigational difficulties, agents mutate and receive rules from other agents through cooperation. The swarm navigates while maintaining a cohesive formation, where shape and size of the formation may vary during navigation, (Hettiarachchi *et al*. 2008) towards a goal through obstacles with their perception obstructed by obstacles.

This paper explores 1) how agent-agent interaction such as receiving rules and mutation rate from another agent during navigation and 2) receiving mean mutation rate computed using agent's better performing neighbors' mutation rates when agents are at a goal affect swarm survival (i.e. agents reaching a goal within a predefined time interval), and presents a detail analysis showing how DAEDALUS can be used to improve swarm survival in an online partially observable environment.

The rest of this paper is organized as follows. Section I part A introduces the DAEDALUS paradigm and part B introduces the partial observability or "Obstructed Perception". Section II describes the artificial physics frame work. Section III presents the experimental methodology with the proposed swarm survival strategy. Section IV provides an analysis of the results, and section V summarizes the results. Section VI discusses the related work in swarm obstacle avoidance and cooperative learning. Finally, concluding

remarks and future work follow in Section VI.

1.1 DAEDALUS

With the DAEDALUS paradigm, we assume that agents (whether software or hardware) move throughout some environment. As they move, they interact with other agents. These agents may be of the same species or of some other species (Spears 1994). Agents of different species have different roles in the environment. The goal is to evolve agent behaviors and interactions between agents, in a distributed fashion, such that the desired global behavior occurs. Let us further assume that each agent has some procedure to control its own actions in response to environmental conditions and interactions with other agents. The precise implementation of these procedures is not relevant, thus they may be programs, rule sets, finite state machines, real-valued vectors, force laws, or any other procedural representation. Agents have a sense of self-worth or "fitness".

Each agent of the swarm is an individual in a population that interacts with its neighbors. Each agent contains a *slightly mutated* copy of the optimized control procedure found with offline learning with an offline EA. This ensures that our agents are not completely homogeneous. We allowed this slight heterogeneity because when the environment changes, some mutations perform better than others. The agents that perform well in the environment will have higher fitness than the agents that perform poorly. When low fitness agents encounter high fitness agents, the low fitness agents ask for the high fitness agent's rules. Hence, better performing agents share their knowledge with their poorer performing neighbors. To ensure the capability of adapting to further changes in the environment, agents also occasionally mutate their own rules, according to a predefined mutation rate attached to that agent. In our original version of DAEDALUS, the agents do not receive mutation rates when they receive the rules.

1.2 Obstructed Perception

When an agent can not see another agent, due to the presence of obstacles, we call this "obstructed perception." When the agent's line of sight lies along an edge of an obstacle, the agents are capable of sensing each other. Figure 1 shows an example scenario of "obstructed perception". The larger circle represents an obstacle, and A and B are agents. We define $minD$ to be the minimum distance from the center of the obstacle to the line of sight of agent A and agent B, and r is the radius of an obstacle. If $r > minD$, the agent A and agent B have their perception obstructed.

We utilize a parameterized description of a line segment (Haeck 2002) to find the $minD$.

$$term_1 = (((1-q) * X_a + q * X_b) - X_c)^2$$

$$term_2 = (((1-q) * Y_a + q * Y_b) - Y_c)^2$$

$$minD = \sqrt{[term_1 + term_2]} \quad (1)$$

where X_a, X_b are the x positions of agents A and B, Y_a, Y_b are the y positions of agents A and B, X_c and Y_c are the x and y positions of the center of an obstacle, and q is the minimum function that is defined by

$$\frac{((X_c - X_a) * (X_b - X_a) + (Y_c - Y_a) * (Y_b - Y_a))}{\left((X_b - X_a)^2 + (Y_b - Y_a)^2\right)} \quad (2)$$

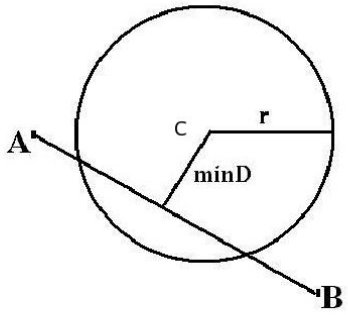

Figure 1: Sensing capability of two agents (A, B) is obstructed by a large obstacle (C).

1.3 Obstacle Avoidance

In prior work (Spears *et al*. 2005) have shown how their artificial physics framework can be used to self-organize swarms of mobile agents into hexagonal lattices (networks) that move towards a goal (see Figure 2). We extended the framework to include motion towards a goal through an obstacle field. An offline EA evolved an agent-level force law, such that agents maintained network cohesion, avoided the obstacles, and reached the goal. The emergent behavior was that the collective moved as a viscous fluid (Hettiarachchi and Spears 2005). In our prior work obstacles did not obstruct perception. The addition of "obstructed perception" makes the task far more difficult, especially as obstacle size increases.

2. The Artificial Physics Framework

In our artificial physics (AP) framework, virtual physics forces drive a swarm of agents to a desired configuration or state. The desired configuration is one that minimizes overall system potential energy, and the system acts as a molecular dynamics ($\vec{F} = m\vec{a}$) simulation.

Each agent has position $\vec{p}$ and velocity $\vec{v}$. We use a discrete-time approximation of the continuous behavior of the agents, with time step Δt. At each time step, the position of each agent undergoes a perturbation $\Delta\vec{p}$. The perturbation depends on the current velocity, i.e., $\Delta\vec{p} = \vec{v}\Delta t$. The velocity of each agent at each time step also changes by $\Delta\vec{v}$. The change in velocity is controlled by the force on

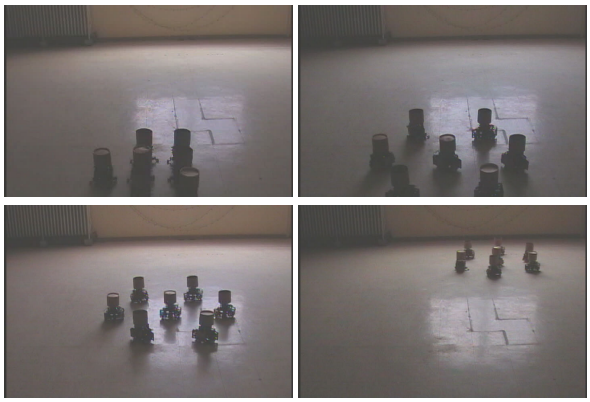

Figure 2: Seven robots form a hexagon, and move towards a light source.

the agent, i.e., $\Delta\vec{v} = \vec{F}\Delta t/m$, where m is the mass of that agent and $\vec{F}$ is the force on that agent. F and v denote the magnitude of vectors $\vec{F}$ and $\vec{v}$. A frictional force is included, for self-stabilization.

From the start, we wished to have our framework map easily to physical hardware, and our model reflects this design philosophy. Having a mass m associated with each agent allows our simulated agents to have momentum. Agents need not have the same mass. The frictional force allows us to model actual friction, whether it is unavoidable or deliberate, in the real robotic system. With full friction, the agents come to a complete stop between sensor readings and with no friction the agents continue to move as they sense. The time step Δt reflects the amount of time the agents need to perform their sensor readings. If Δt is small, the agents get readings often, whereas if the time step is large, readings are obtained infrequently. We have included a parameter F_{max}, which provides a necessary restriction on the acceleration a agent can achieve. Also a parameter V_{max} restricts the maximum velocity of the agents (and can always be scaled appropriately with Δt to ensure smooth path trajectories).

In this paper we utilize a generalized Lennard-Jones (LJ) force law as the control procedure of our agents. The LJ potential function models two distinct forces between neutral molecules and atoms. The forces are based on the distances between the molecules; at long ranges the attractive force makes the molecules move closer and at short ranges the repulsive force makes the molecules move apart, causing the molecules to maintain a natural balance. We derive the force function (i.e. negated derivative of the potential function) for interaction between two agents as:

$$F_{i,j} = 24\epsilon\left[\frac{2d\sigma^{12}}{r^{13}} - \frac{c\sigma^{6}}{r^{7}}\right] \quad (3)$$

$F_{i,j} \leq F_{max}$ is the magnitude of the force between two agents i and j, and r is the distance between the two agents. σ is the desired separation between agent i and agent j (i.e. all other neighboring agents). The variable ϵ affects the strength of the force, while c and d control the relative balance between the attractive and repulsive components. In order to achieve optimal behavior, the values of ϵ, c, d, and F_{max} must be determined. Our motivation for using the LJ force law is that (depending on the parameter settings) it can easily model crystalline solid formations, liquids, and even gases.

In the LJ potential function shown in Figure 3, whenever $\epsilon = 1$ and $\sigma = r$, the interaction energy between two molecules is at zero, which is the molecule's equilibrium. When the separation distance $r > 1$, interaction energy quickly decreases to -1 and then increases and eventually reaches zero due to longer range, causing non-interaction between molecules. When $r < 1$, the interaction energy between two molecules is very high, reaching ∞. Due to the behavior shown by the LJ potential function, this becomes an ideal function to model interactions between agents and their environments.

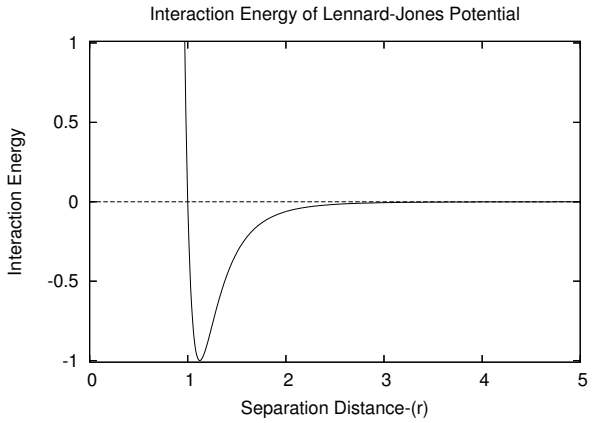

Figure 3: Interaction energy of LJ with $\epsilon = 1$ and $\sigma = 1$.

3. Experimental Methodology

To achieve the best performance, we have separate force laws for agent-agent interactions, agent-goal interactions, and agent-obstacle interactions. Hence ϵ, c, d, and F_{max} must be optimized offline for all three forms of interactions, resulting in 12 parameters. Agent-agent and agent-obstacle interactions are local (i.e., agents can only sense nearby agents and obstacles). The agents are trained with an offline EA, in an offline environment. The agents in offline environment do not make use of DAEDALUS paradigm.

Offspring are generated using one-point crossover with a crossover rate of 60%. Mutation adds/subtracts an amount drawn from a $N(0, \delta)$ Gaussian distribution. Each parameter has a $1/L$ probability of being mutated, where L is the length of the individual. Mutation ensures that parameter values stay within accepted ranges.

Since we are using an EA that minimizes, the offline performance of an individual is measured as a weighted sum of penalties:

$$w_1 P_{Collision} + w_2 P_{NoCohesion} + w_3 P_{NotReachGoal}$$

The weighted multi-objective fitness function consists of three criteria: collision avoidance, maintaining cohesion, and reaching the goal within a time limit. At each time step, each criteria is evaluated and added a penalty for under performance. Since there is no safety zone around the obstacles (Balch and Hybinette 2000), a penalty is added to the score if the agents collide with obstacles. The cohesion penalty is derived from the fact that in a good hexagonal lattice, interior agents should have six local neighbors. A penalty occurs if an agent has more or less neighbors. If no agent reaches the goal within the time limit, a penalty occurs.

The environment, as shown in Figure 4, is 900 $\times$ 700 with 90 randomly positioned obstacles and each of radius 10, where are all units are in pixels. This yields about 4.5% obstacle coverage, which is typical of most studies in this area (Balch and Hybinette 2000). The agents move with maximum velocity 20 units/sec. The EA does not have great difficulty producing an optimized LJ force law that avoids obstacles while allowing all agents to reach the goal.

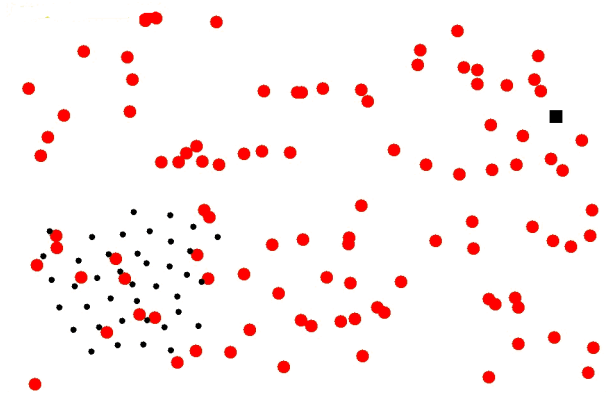

Figure 4: 40 agents moving to the goal in offline environment. The larger circles represent obstacles, while the square in the upper right represents the goal.

However, the online environment is far more difficult. The online 2D world is 1000 $\times$ 850, and each of the 90 obstacles has a radius of 20 compared to the offline obstacle radius of 10, where all units are in pixels again. Therefore more than 13.3% of the online environment is covered with the obstacles, nearly tripling the obstacle density. We also increase the maximum velocity of the agents to 30 units/sec from 20 units/sec, making the agents move 1.5 times faster than in the offline environment. *"Obstructed perception" occurs in both the offline and online environments.*

For the online environment, each agent of the swarm contains a slightly mutated copy of the optimized LJ force law rule set found with offline learning. There are five goals to achieve in a long corridor, and between each randomly positioned goal is a different obstacle course with 90 randomly positioned obstacles. The LJ force law learned in offline mode is not sufficient for this more difficult environment, resulting in agents that never reach the goal (due to the high percentage of obstacles).

Agents that are left behind (due to obstacle cul-de-sacs) do not proceed to the next goal, but agents that collide with obstacles and make it to the goal are allowed to proceed to the next goal. We assume that damaged agents can be repaired once they reach a goal. Although the noise in dynamic environments is not specifically modeled in our simulation, it has been shown with actual agents that the Artificial Physics framework is robust to modest amounts of noise (Spears *et al.* 2005). In fact, noise can actually improve performance by overcoming local optima in the behavior space (Martinson and Payton 2005; Spears and Spears 1999).

In our prior work (Hettiarachchi and Spears 2005; Hettiarachchi *et al.* 2006; Hettiarachchi and Spears 2006; Hettiarachchi *et al.* 2008), we have shown that the agents easily learned to avoid colliding with obstacles and maintain cohesive formations, so our focus in this paper is on agent survival (i.e. the number of agents that reach a goal). When the agents are left behind in cul-de-sacs, the number of agents that survive to reach a goal reduces, and this causes the cohesion of the formation to be reduced. We utilized two different approaches to improve swarm survival.

- Approach 1: if agent$_i$ is not moving (due to an obstacle in the way) and a neighboring agent$_j$ is moving, then agent$_i$ receives agent$_j$'s agent-agent interactions, including agent$_j$'s mutation rate.
- Approach 2: if agent$_i$ is at a goal, agent$_i$ computes the mean mutation rate using agent$_i$'s better performing neighbors' mutation rates. The agent$_i$ only consider the neighbors who have higher self-worth or "fitness" than the agent$_i$. Then the agent$_i$ uses this mean mutation rate as its own before moving to next goal.

4. Results and Analysis

We compared all our results of DAEDALUS to three control studies. In the first control study, we train the agents with an offline EA on small obstacles, and then test them again on small obstacles to verify their performance. In the second control study, we train the agents with an offline EA on large obstacles and test them on large obstacles. The purpose of this control study is to clarify the difficulty of the task. Finally, in the third control study, we train the agents with an offline EA on small obstacles and test them on large obstacles. The purpose of this study was to see how well the knowledge learned while avoiding small obstacles transferred when navigating in environments with large obstacles. The results of all control studies presented in this paper are averaged over 25 independent runs.

The Figure 5 shows the results of our first experiment where the two approaches explained in the Section 3. are not utilized. In all the figures presented in this section, the y-axis shows the number of agents that survive to reach a goal (there are five goals) at each stage of the long corridor. The top performance curve is for the first control study. Note that learning with small obstacles in the offline mode is not hard, and the agents perform very well in the online environment. This is due to the fact that the small obstacles make the environment less dense providing the agents

sufficient space to navigate. Out of the 60 initial agents released into the online environment, 93.3% survived to reach the last goal. With such small obstacles (which is the maximum density examined in majority of the related literature), the "obstructed perception" is not an important issue.

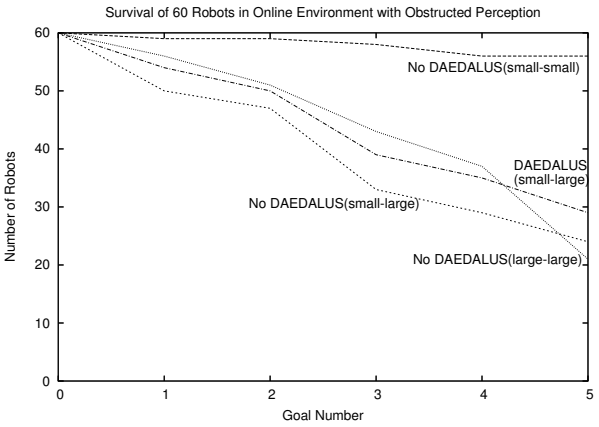

Figure 5: The number of agents that survive to reach a goal. Agents are not allowed to receive mutation rates from another agent during navigation. Agents do not receive mean mutation rate at a goal.

For the DAEDALUS performance curve labeled "DAEDALUS(small-large)", we divided the agents into five groups of equal size. Each group of 12 agents was assigned a mutation rate of 1%, 3%, 5%, 7%, and 9%, respectively. In this control study, agents are not allowed to receive mutation rate from another agent and the agents do not receive mean mutation rate of its neighbors at a goal. Out of the initial 60 agents, 29 or 48.3% agents survived to reach the final goal.

Since the mutation rate has a major effect on the swarm performance (Hettiarachchi and Spears 2006), we decided to explore the mutation rates of the five groups. The Table 1 shows the number of agents that reach a goal and their mutation makeup. Out of the 29 agents that survived to reach the final goal, 9 agents or 31% had 5% mutation rate. Only 3 agents or about 10% of agents with a mutation rate of 9% survived to reach the final goal. The agents with 3% and 9% mutation rates had difficulty coping with the changed environment; the performance of the agents with 1% and 7% mutation rates is similar to the agents with 5% mutation rate.

In an attempt to improve the survivability, we again initialized the five agent groups as in previous experiment with the mutation rates of 1%, 3%, 5%, 7%, and 9%, respectively. However, in this second experiment, if an agent receives rules from a neighbor (which, again, occurs if that agent is in trouble, agents getting stuck behind cul-de-sacs is an example), it also receives the neighbor's mutation rate. This is the first approach listed in the Section 3. In this experiment, the agents do not receive mean mutation rate of its neighbors at a goal. The curve with the label "DAEDALUS(small-large)" in Figure 6 shows the results of this study.

Out of the initial 60 agents, 37 or 61.6% agents survived

	Mutation Rate				
agents survive	1%	3%	5%	7%	9%
60-start	12	12	12	12	12
54-goal 1	11	11	11	11	10
50-goal 2	10	9	11	10	10
39-goal 3	8	6	10	9	6
35-goal 4	8	5	9	8	5
29-goal 5	6	4	9	7	3

Table 1: The number of agents that survive to reach a goal and their mutation rates. Agents are not allowed to receive mutation rates from another agent during navigation. Agents do not receive mean mutation rate at a goal.

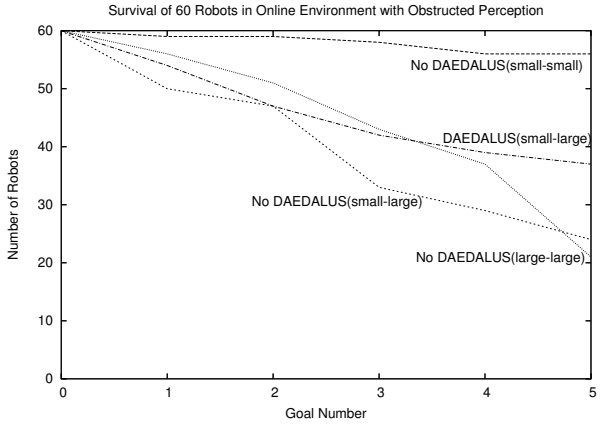

Figure 6: The number of agents that survive to reach a goal. Agents are allowed to receive mutation rates from another agent during navigation. Agents do not receive mean mutation rate at a goal.

to reach the final goal. The number of agents reaching the final goal is an 21.6% improvement over the results seen in Figure 5, where we do not allow the agents to receive another agent's mutation rate. Though this improvement is not statistically significant, it provides a practical significance to various swarm applications.

Table 2 shows the number of agents that reach a goal and their mutation makeup for the second experiment. In this implementation, agents in trouble not only change their rules, but their mutation rate. The agents do not receive the mean mutation rate of its neighbors at a goal.

Once again, it is clear that the agents favor 5% and 7% mutation rates over 1%, 3%, and 9%. Interestingly, this adaptation shown in Table 2 is not rapid enough to create a significant impact on mutation makeup of the swarm. Due to "obstructed perception", the agent-agent cooperation is limited during navigation. This lack of cooperation limits the agents ability to make informed decisions about better performing mutation rates.

Our results suggest that for the agents to make informed decisions on good mutation rates to be received, an agent in trouble is required to cooperate with a large number of

	Mutation Rate				
agents survive	1%	3%	5%	7%	9%
60-start	12	12	12	12	12
54-goal 1	10	10	11	12	11
47-goal 2	9	8	11	10	9
42-goal 3	6	7	11	10	8
39-goal 4	6	6	10	9	8
37-goal 5	5	6	10	9	7

Table 2: The number of agents that survive to reach a goal and their mutation rates. Agents are allowed to receive mutation rates from another agent during navigation. Agents do not receive mean mutation rate at a goal.

other agents. This requirement has limitations during navigation due to "obstructed perception" and the varying formation cohesiveness. Due to the behavior of the LJ force law, the fluid like motion learned by the swarm causes the swarm formation to stretch, reducing cohesiveness when navigating around the obstacles. When the agents are at a goal, "obstructed perception" is minimal due to lack of obstacles around a goal, and the swarm cohesion is high due to agent-goal interaction. We take these observations into consideration in the second approach explained in the Section 3. In the second approach, agents receive mean mutation rate of its neighbors at a goal.

When the swarm is at a goal, each agent computes the mean mutation rate using that agent's better performing neighbors' mutation rates. The neighbor of an agent is defined by the agent's sensor range, where $r \leq \sigma$ distance. Each agent accumulate the mutation rates of the neighbors who have better self-worth or "fitness", and computes the mean mutation rate. The agent then uses this mean mutation rate to navigate to the next goal. In this experiment, all agents compute the mean mutation rate at each goal in the long corridor with five goals, and the agents that are in trouble continue to receive mutation rates from other agents. This experiment combines the two approaches presented in the Section 3. The curve with the label "DAEDALUS(small-large)" in Figure 7 shows the results of this experiment.

Out of the initial 60 agents, 41 or 68.3% agents survived to reach the final goal. This is statistically as well as practically significant gain over the results in Figure 5.

The results in Table 3 clearly show that the swarm favors mutation rates in two ranges, $3\% < mutation\ rate \leq 5\%$ and $5\% < mutation\ rate \leq 7\%$ over the other three mutation rate ranges. The agents that uses the mean of better performing neighbors' mutations at a goal rapidly adapt to their online environment causing the swarm to maintain the agent survival over the corridor with five goals.

This approach does not introduce new global knowledge in to our swarm system, and allows the swarm behavior to emerge through local interactions. The results suggest that "obstructed perception" has a significant impact on swarm survival. The "obstructed perception" limits the agent's ability to receive mutation rate from another agent during navigation, but once the swarm is at a goal, the agents have

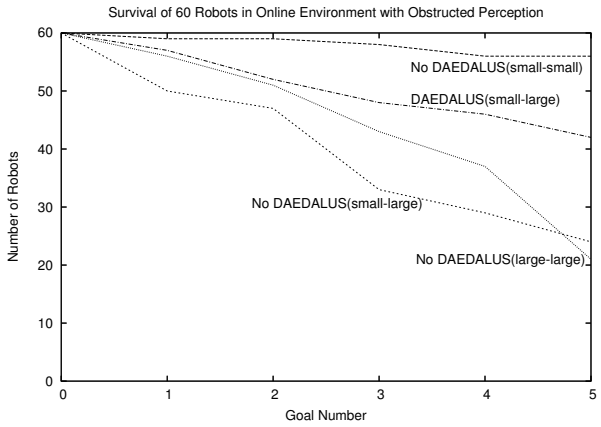

Figure 7: The number of agents that survive to reach a goal. Agents are allowed to receive mutation rates from another agent. Agents receive mean mutation rate at a goal.

	Mutation Rate Ranges				
agents survive	$\leq 1\%$	$\leq 3\%$	$\leq 5\%$	$\leq 7\%$	$\leq 9\%$
60-start	12	12	12	12	12
54-goal 1	11	11	11	11	10
51-goal 2	1	1	27	22	0
48-goal 3	0	1	30	17	0
45-goal 4	0	1	27	17	0
41-goal 5	0	2	24	15	0

Table 3: The number of agents that survive to reach a goal and their mutation rates. Agents are allowed to receive mutation rates from another agent during navigation. Agents receive mean mutation rate at a goal.

improved access to their neighbors due to the swarm cohesion. This cohesive formation at a goal increases the number of neighbors each agent encounters and allows higher agent-agent interactions.

To better understand the impact of the combined approaches presented in Figure 7, we conducted an experiment only with the second approach presented in the Section 3. In this experiment, the agents compute and use the mean mutation rate at each goal, but not receive the mutation rates from other agents during navigation. The results of this experiment is shown in Figure 8.

Out of the initial 60 agents, 34 or 56.6% agents survived to reach the final goal. The outcome is quite similar to the second control study in Figure 6, where 37 agents survived to reach the final goal. Again, there is no significant gain in survival if only one of the approaches presented in the Section 3. is used.

The results in Table 4 clearly show that the swarm again favors mutation rates in two ranges, $3\% < mutation\ rate \leq 5\%$ and $5\% < mutation\ rate \leq 7\%$ over the other three mutation rate ranges. Utilizing a single approach do not improve the survival, but utilizing the two approaches together improve swarm survival significantly.

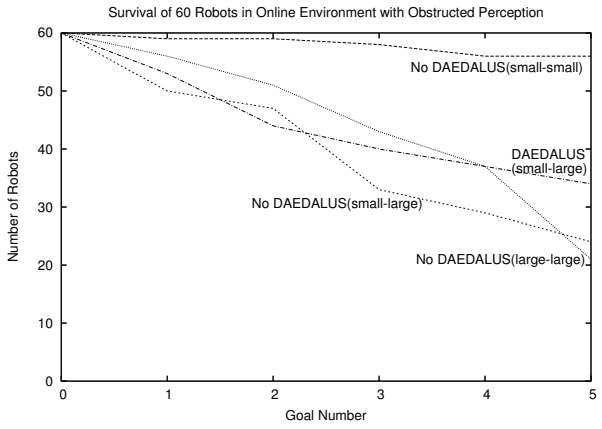

Figure 8: The number of agents that survive to reach a goal. Agents are not allowed to receive mutation rates from another agent. Agents receive mean mutation rate at a goal.

	Mutation Rate Ranges				
agents survive	$\leq 1\%$	$\leq 3\%$	$\leq 5\%$	$\leq 7\%$	$\leq 9\%$
60-start	12	12	12	12	12
53-goal 1	11	10	11	10	11
44-goal 2	0	1	17	26	0
40-goal 3	0	0	16	24	0
37-goal 4	0	0	13	24	0
34-goal 5	0	0	17	17	0

Table 4: The number of agents that survive to reach a goal and their mutation rates. Agents are not allowed to receive mutation rates from another agent. Agents receive mean mutation rate at a goal.

5. Summary

Surviving in one environment with the force laws learned in an another environment is difficult. This difficulty further increases with the introduction of "obstructed perception". In the offline simulation the agents did not learn to avoid cul-de-sacs; the obstacle density did not produce cul-de-sacs, leaving sufficient space for the agents to navigate through. In the online environment agents are not capable of completely avoiding cul-de-sacs, so they get stuck behind these cul-de-sacs. To improve swarm survival, we utilized two approaches. The first approach allows the agents to receive rules and mutation rates from other agents during navigation and the second approach allows the agents to compute and use the mean mutation rate of the neighborhood when the swarm is at a goal. The least survival rate occurs when the two approaches are not utilized, utilizing just one approach improves the survival rate to an insignificant level, and utilizing the two approaches together improves the survival rate significantly.

6. Related Work

Most of the swarm literature can be subdivided into *swarm intelligence*, *behavior-based*, *rule-based*, *control-theoretic* and *physics-based* techniques. Swarm intelligence techniques are ethologically motivated and have had excellent success with foraging, task allocation, and division of labor problems (Bonabeau *et al*. 1999; Hayes *et al*. 2001). Both behavior-based and rule-based systems (Fredslund and Matarić 2002; Balch and Arkin 1998; Schultz and Parker 2002) have proven quite successful in demonstrating a variety of behaviors in a heuristic manner. Behavior-based and rule-based techniques do not make use of potential fields or forces. Instead, they deal directly with velocity vectors and heuristics for changing those vectors (although the term "potential field" is often used in the behavior-based literature, it refers to a field that differs from the strict Newtonian physics definition). Control-theoretic approaches have also been applied effectively (e.g., (Fax and Murray 2002)). Our approach does not make the assumption of having leaders and followers, as in (Desai *et al*. 1998; 2001).

In the specific context of obstacle avoidance, the most relevant papers are (Balch and Arkin 1998; Balch and Hybinette 2000; Fredslund and Matarić 2002). Balch (Balch and Arkin 1998) examines the situation of four agents moving in formation through an obstacle field with 2% coverage. In (Balch and Hybinette 2000) he extends this to an obstacle field of 5% coverage, and also investigates the behavior of 32 agents moving around one medium size obstacle. Fredslund and Matarić (Fredslund and Matarić 2002) examine a maximum of eight agents moving around two wall obstacles.

The work done in (OHara *et al*. 2005) uses an embedded network distributed throughout the environment to approximate the path-planning space and use the network to compute a navigational path using GNATs when the environment changes. The dynamism of the environment is modeled with an opening and closing door in the experimental setup. However, the embedded network is immobile, whereas our network is completely mobile.

The work done in (Tan 1993) uses reinforcement learning to address agent learning by sharing instantaneous information, episodes, and learned policies. The task environment is a 10 by 10 grid world with maximum of 4 agents, 2 hunters and 2 prey. Our work is conceptually similar and was developed independently. The cooperative learning discussed in (Tan 1993) is fundamentally offline, whereas in our approach learning is both offline and online where agents continue to adapt to their changing environment through cooperation. The agents depending only on offline learning approach can be problematic due to complex nature and the unexpected changes in the environment.

7. Conclusion and Future Work

In this paper, we explore how agent-agent interaction such as receiving rules and mutation rate from another agent during navigation affect swarm survival (i.e. agents reaching a goal within a predefined time interval). To overcome navigational difficulties, agents mutate and receive rules from other agents through cooperation, while navigating through obstacles towards a goal with "obstructed perception", also maintaining their formation. We explored two approaches 1) how

agent-agent interaction such as receiving rules and mutation rate from another agent during navigation and 2) receiving mean mutation rate of agent's neighbors when agents are at a goal affect swarm survival. The results suggest that obstructed perception has a significant impact on swarm survival, but the agents can make use of higher level of swarm cohesion at a goal to overcome the difficulty of finding sufficient amount of "worthy" neighbors. We presented a detail analysis of the two approaches showing how DAEDALUS can be used to improve swarm survival in an online environment.

Future work of this research will focus on the issue of credit assignment, when fitness feedback is sporadic. Current work in classifier systems uses mechanisms such as "bucket-brigade" or "profit sharing" to allocate rewards to individual agents appropriately (Grefenstette 1988). However, these techniques rely on global blackboards and assume that all agents can potentially act with all others through a bidding process. We intend to modify these approaches so that they are fully distributed and appropriate for online systems.

8. Acknowledgments

Author is especially grateful to Dr. William M. Spears at Swarmotics LLC. for his valuable comments and suggestions.

References

Balch, T. and Arkin, R. (1998). Behavior-based Formation Control for Multi-Robot Teams. *IEEE Transactions on Robotics and Automation,* **14**: 1–15.

Balch, T. and Hybinette, M. (2000). Social Potentials for Scalable Multi-Robot Formations. *Proceedings of the IEEE International Conference on Robotics and Automation,* **1**: 73–80.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. *Santa Fe Institute Studies in the Sciences of Complexity, Oxford University Press.*

Desai, J., Ostrowski, J. and Kumar, V. (1998). Controlling Formations of Multiple Mobile Robots. *IEEE International Conference on Robotics and Automation,* **4**: 2864–2869.

Desai, J., Ostrowski, J. and Kumar, V. (2001). Modeling and Control of Formations of Nonholonomic Mobile Robots.*IEEE Transactions on Robotics and Automation,* **17**: 905–908.

Fax, J. and Murray, R. (2002). Information Flow and Cooperative Control of Vehicle Formations. *IEEE Transactions on Automatic Control,* **49**: 1465–1476.

Fredslund, J. and Matarić, M. (2002). A General Algorithm for Robot Formations Using Local Sensing and Minimal Communication. *IEEE Transactions on Robotics and Automation,* **18**: 837–846.

Grefenstette, J. (1988). Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms. *Machine Learning Journal, Springer,* **3**: 225–245.

Grefenstette, J. (1989). A System for Learning Control Strategies with Genetic Algorithms. *Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann,* 183–190.

Haeck, N. (2002). Minimum Distance Between a Point and a Line. `http://www.simdesign.nl/tips/tip001.html`

Hayes, A., Martinoli, A. and Goodman, R. (2001). Swarm Robotic Odor Localization. *IEEE/RSJ International Conference on Intelligent Robots and Systems,* **21**: 427–441.

Hettiarachchi, S. and Spears, W. (2005). Moving Swarm Formations Through Obstacle Fields. *International Conference on Artificial Intelligence, CSREA Press,* **1**: 97–103

Hettiarachchi, S., Spears, W., Green, D. and Kerr, W. (2006). Distributed Agent Evolution with Dynamic Adaptation to Local Unexpected Scenarios. *Proceedings of the 2005 Second GSFC/IEEE Workshop on Radical Agent Concepts, Springer,* **3825**: 245–256.

Hettiarachchi, S. and Spears, W. (2006). DAEDALUS for Agents with Obstructed Perception. *Proceedings of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems, IEEE Press,* 195–200.

Hettiarachchi, S.; Maxim, P.; Spears, W.; and Spears, D., (2008). Connectivity of Collaborative Robots in Partially Observable Domains. *Proceedings of the International Conference on Control, Automation and Systems, Seoul, Korea, IEEE Press,* 721–728.

Martinson, E.; and Payton, D. (2005). Lattice Formation in Mobile Autonomous Sensor Arrays. *Lecture Notes in Computer Science,* **3342**: Springer, 98–111.

O'Hara, K. J., Bigio, V. L., Dodson, E. R., Irani, A., Walker, D. B. and Balch, T. R. (2005). Physical Path Planning Using the GNATs. *Proceedings of the IEEE International Conference on Robotics and Automation,* 709–714.

Schultz, A. and Parker, L. (2002). Multi-Robot Systems: From Swarms to Intelligent Automata. *Proceedings of the First International Workshop on Multi-robot Systems,* 73–82.

Spears, W. (1994). Simple Subpopulation Schemes. *Proceedings of the Evolutionary Programming Conference, World Scientific,* 296–307.

Spears, W. and Spears, D. (1999). Using Artificial Physics to Control Agents. *Proceedings of the IEEE International Conference on Information, Intelligence, and Systems,* 281–288

Spears, W., Spears, D., Hamann, J. and Heil, R. (2005). Distributed, Physics-Based Control of Swarm of Vehicles. Autonomous Robots, *Kluwer,* **17**: 137–164.

Tan M. (1993). Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. *Proceedings of the Tenth International Conference on Machine Learning,* Morgan Kaufmann, 330–337.

Wu, A., Schultz, A. and Agah, A. (1999). Evolving Control for Distributed Micro Air Vehicles. *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, IEEE Press,* 174–179.