International Journal of Intelligent Computing and Cybernetics
© Emerald Group Publishing Limited

DISTRIBUTED ADAPTIVE SWARM FOR OBSTACLE AVOIDANCE

SURANGA HETTIARACHCHI

Computer Science Department
Indiana University Southeast
4201, Grant Line Road
New Albany, Indiana, 47150, USA
suhettia@iu.edu
http://www.ius.edu/suhettia

WILLIAM M. SPEARS

Swarmotics, LLC
Laramie, WY 82070, USA
wspears@swarmotics.com
http://www.swarmotics.com

Received (06 February 2009)
Revised (12 August 2009)
Accepted (18 August 2009)

Abstract

Purpose- This paper demonstrates a novel use of a generalized Lennard-Jones (LJ) force law in *Physicomimetics*, combined with offline evolutionary learning, for the control of swarms of robots moving through obstacle fields towards a goal. We then extend the paradigm to demonstrate the utility of a real-time online adaptive approach named DAEDALUS.

Design/Methodology/Approach- To achieve the best performance, we optimize the parameters of the force law used in our *Physicomimetics* approach, using an evolutionary algorithm (offline learning). We utilize a weighted fitness function consists of three components: a penalty for collisions, a penalty for lack of swarm cohesion, and a penalty for robots not reaching the goal. We then give each robot of the swarm a slightly mutated copy of the optimized force law rule set found with offline learning and introduce the robots to a more difficult environment. We use our online learning framework (DAEDALUS) for swarm adaptation in this more difficult environment.

Findings- The novel use of the generalized Lennard-Jones (LJ) force law combined with an evolutionary algorithm surpasses the prior state-of-the-art in the control of swarms of robots moving through obstacle fields. In addition, our DAEDALUS framework allows the swarms of robots to not only learn and share behavioral rules in changing environments (in real time), but also to learn the proper amount of behavioral exploration that is appropriate.

Research limitations/implications- There are significant issues that arise with respect to ”wall following methods” and ”local minimum trap” problems. We have observed ”local minimum trap” problems in our work, but we did not address this issue in detail. We intend to explore other approaches to develop more robust adaptive algorithms for online learning. We believe that we can accelerate the learning of the proper amount of behavioral exploration.

Practical implications- In order to provide meaningful comparisons, we provide a more complete set of metrics than prior papers in this area. We examine the number of collisions between robots and obstacles, the distribution in time of the number of robots that reach the goal, and the connectivity of the formation as it moves.

Originality/value- We address the difficult task of moving a large number of robots in formation through a large number of obstacles. The important real-world constraint of “obstructed perception” is modeled. The obstacle density is approximately three times the norm in the literature. We show how concepts from population genetics can be used with swarms of agents to provide fast online adaptive learning in these challenging environments. In addition, this paper also presents a more complete set of metrics of performance.

Keywords: *Physicomimetics*; obstacle avoidance; adaptive learning; obstructed perception; DAEDALUS.

Paper type: **Research paper**

1. Introduction

The focus of our research is to design and build rapidly deployable, adaptive, cost-effective, and autonomous distributed robot swarms. Our objective is to provide a scientific, yet practical, approach to the design and analysis of swarm behaviors.

The team of robots could vary widely in type, as well as size, e.g., from nanobots to micro-air vehicles (MAVs) and micro-satellites. A robot’s sensors perceive the world, including other robots, and a robot’s effectors make changes to that robot and/or the world, including other robots. It is assumed that robots can only sense and affect nearby robots; thus, a key challenge has been to design ”local” control rules. Not only do we want the desired global behavior to emerge from the local interaction between robots (self-organization), but we also require fault-tolerance, that is, the global behavior degrades very gradually if individual robots are damaged. Self-repair is also desirable, in the event of damage. Self-organization, fault-tolerance, and self-repair are precisely those principles exhibited by natural physical systems. Thus, many answers to the problems of distributed control can be found in the natural laws of physics.

In this paper we focus on the application of *Physicomimetics* to swarms of robots moving through obstacle fields [Hettiarachchi and Spears (2005)]. Our objective was two-fold. Prior research in this area has generally focused either on a small number of robots moving through a large number of obstacles, or a large number of robots moving through a small number of obstacles [Balch and Arkin (1998); Balch and Hybinette (2000)]. However, the more difficult task of moving a large number of robots in formation through a large number of obstacles is generally not addressed. Also, proposed metrics of performance are not complete, ignoring criteria such as the number of collisions between robots and obstacles, the distribution in time of the number of robots that reach the goal, and the connectivity of the formation as it moves. Hence, one objective was to provide a more complete set of metrics from which meaningful comparisons could be made. Second, we used these metrics, coupled with a more complete experimental methodology, to examine (a) different strategies for performing the task, and (b) trade-offs between different criteria.

Several other issues must be addressed before swarms of robots can be successfully deployed. Due to numerous constraints, such as noise, limited range of interaction with other agents, delayed feedback, and the distributed autonomy of the agents, it is difficult to engineer swarms that learn and adapt in real time [Grefenstette (1989); Wu *et al*. (1999)].

In traditional "off-line" approaches, an evolutionary algorithm (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 this 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 versus homogeneity).

In this paper we also introduce a novel paradigm for swarm adaptive learning in online called "DAEDALUS"and show how DAEDALUS can be used to achieve global aggregate behavior of agents that move through an obstacle field towards a goal when the obstacles obstruct the perception of the agents (i.e. they act to degrade the interactions between agents).

2. *Physicomimetics* Framework

This section provides a brief overview of the *Physicomimetics* framework for distributed control of robots in a swarm [Spears and Spears (1999)]. In the *Physicomimetics* framework, virtual physics forces drive a swarm robotics system 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 robot has position $\vec{p}$ and velocity $\vec{v}$. We use a discrete-time approximation to the continuous behavior of the robots, with time-step Δt. At each time step, the position of each robot 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 robot at each time step also changes by $\Delta\vec{v}$. The change in velocity is controlled by the force on the robot, i.e., $\Delta\vec{v} = \vec{F}\Delta t/m$, where m is the mass of that robot and $\vec{F}$ is the force on that robot. F and v denote the magnitude of vectors $\vec{F}$ and $\vec{v}$. A frictional force is included for self-stabilization and is modeled by decreasing the robot's velocity by a constant multiplicative factor (< 1) at each time step. Figure 1 shows the perturbation of the robots R and R_4 due to forces exerted upon them by other robots and the environment.

Our objective is to have the *Physicomimetics* framework map easily to physical hardware, and the *Physicomimetics* framework reflects this design philosophy. Having a mass m associated with each robot allows our simulated robots to have momentum. Robots 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 robots come to a complete stop between sensor readings and with no friction the robots continue to move as they sense. The time step Δt reflects the amount of time the robots need to perform their

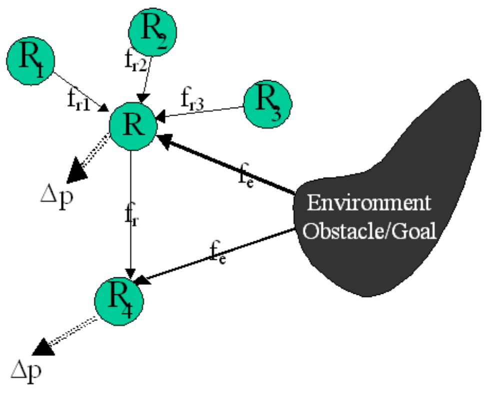


Fig. 1. Robots R and R_4 undergo a perturbation to their positions due to forces from other robots and the environment. Robot R_4 does not sense forces from robots R_1 through R_3 due to sensor proximity.

sensor readings. If Δt is small, the robots get readings very often whereas if the time step is large, readings are obtained infrequently. We have also included a parameter F_{max}, which provides a necessary restriction on the acceleration a robot can achieve. Also, a parameter V_{max} restricts the maximum velocity of the robots (and can always be scaled appropriately with Δt to ensure smooth path trajectories). Figure 2 (top) shows seven outdoor robots running in formation on Prexy's Pasture at the University of Wyoming (left) and a dirt road (right), using *Physicomimetics*. Figure 2 (bottom) shows the same robots (with the same code) pulling an object with a very uneven distribution of weight and friction.

2.1. *Newtonian Force Law*

The Newtonian Force Law (Newtonian) has been used in prior work [Spears *et al*. (2005)] and is a generalization of the "Newtonian" gravitational force law which includes both attraction and repulsion. The force law is:

$$F_{i,j} = \frac{m_i m_j G}{r^p} \tag{1}$$

$F \leq F_{max}$ is the magnitude of the force between two robots i and j, and r is the distance between the two robots. The masses of the robots are denoted as m_i and m_j, and are assumed to be set to 1.0 in this paper. The variable G affects the strength of the force. The variable p is a user-defined power that controls the reduction in strength with distance. The force is repulsive if $r < R$, attractive if $r > R$, and is zero beyond a certain range (e.g., $1.5R$), to enforce the local nature of the force law. R is the desired separation between a robot and neighboring robots. In order to achieve optimal behavior, the values of G, p, and F_{max} must be determined as well as the amount of friction. The Newtonian force law generally creates rigid formations that act as solids, even in the presence of sensor and locomotion uncertainty.

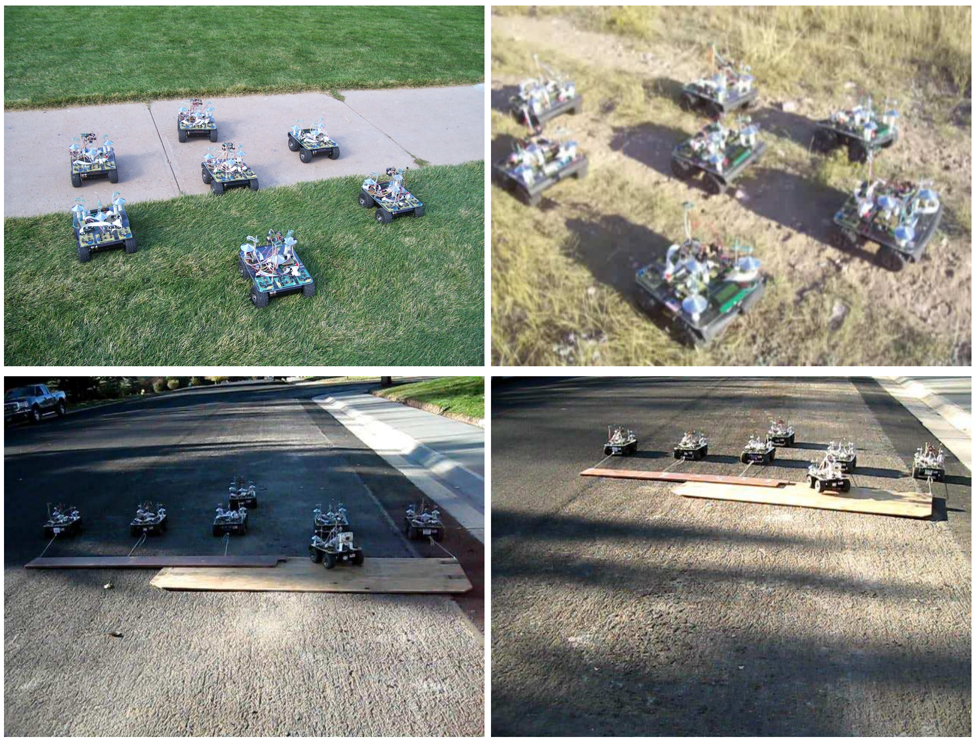


Fig. 2. Seven robots using *Physicomimetics* in various outdoor formation and pulling experiments.

2.2. *Lennard-Jones (LJ) Force Law*

In this paper we also investigate the utility of a second force law, which is a generalization of the Lennard-Jones (LJ) force law. The LJ potential function was first proposed by John Lennard-Jones in 1929. This potential function models two distinct forces between neutral molecules and/or 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. The LJ potential function can be given by the expression:

$$LJP_r = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{2}$$

Whenever $\sigma = r$ the interaction energy between two molecules is at zero. The potential function is shown in Figure 3 with $\varepsilon = 1$ and $\sigma = 1$. When the separation distance $r > 1$, interaction energy quickly decreases to -1 and then increases and eventually reaches zero at longer range, causing non-interaction between molecules. When $r < 1$, the interaction energy between two molecules is very high, reaching ∞. The minimum of the function occurs at $r = 2^{1/6}\sigma$, with a value of $-\varepsilon$. Due to the behavior shown by the LJ potential function, this becomes an ideal function to model interactions between robots and their environments.

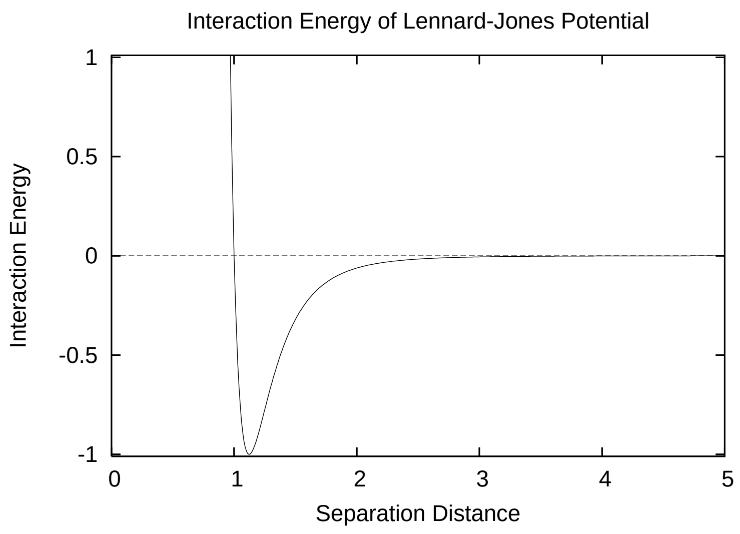


Fig. 3. Interaction potential of LJ with $\varepsilon = 1$ and $\sigma = 1$.

To model interactions of robots in a swarm, we need to transform the LJ potential function to a force function. Since the force between two molecules is the negated derivative of the potential,

$$F = -\left(\frac{d\,(LJP_r)}{dr}\right), \tag{3}$$

the force between robots i, j is:

$$F_{i,j} = \frac{-4\varepsilon}{r}\left[\frac{-12\sigma^{12}}{r^{13}} + \frac{6\sigma^{6}}{r^{7}}\right] \tag{4}$$

and $R = 2^{1/6}\sigma$ is the desired distance between two robots. We generalize the force function for interaction between two robots as:

$$F_{i,j} = 24\varepsilon\left[\frac{2d\sigma^{12}}{r^{13}} - \frac{c\sigma^{6}}{r^{7}}\right] \tag{5}$$

Again, $F \le F_{max}$ is the magnitude of the force between two robots, and r is the distance between the two robots. 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 as well as the amount of friction. Our motivation for trying the LJ force law is that (depending on the parameter settings) it can easily model crystalline solid formations, liquids, and gases. The pseudocode of the *Physicomimetics* algorithm that uses the LJ force law for robot-robot interactions can be found in Hettiarachchi [2007]. By changing the parameter settings of the force law, we can model liquid or solid behavior of the swarm.

In the next section we compare the performance of *Physicomimetics* on an obstacle avoidance task, using the two force laws defined above.

3. Methodology

Our simulation architecture, as shown in Figure 4, consists of four modules: an EA for evolving the population of force laws, an environment generator, a global observer that evaluates the performance of a particular force law, and a performance measurement module that evaluates the quality of the optimum force law. A detailed discussion of the performance measurement module is provided in Section 3.3.

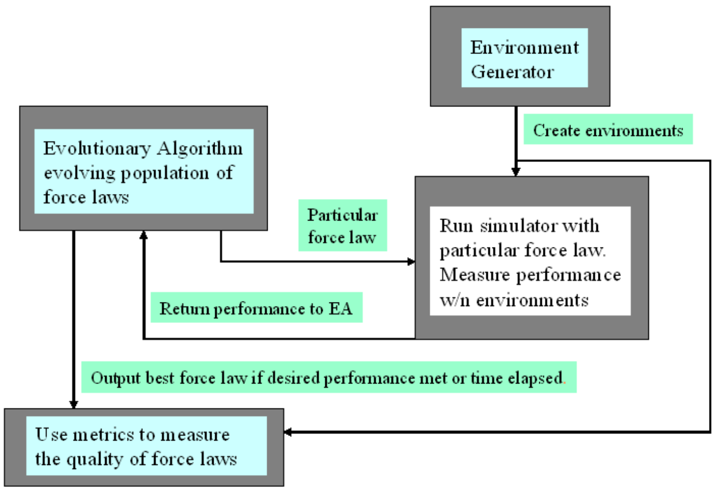

Environment Generator
Create environments
Evolutionary Algorithm evolving population of force laws
Particular force law
Run simulator with particular force law. Measure performance w/n environments
Return performance to EA
Output best force law if desired performance met or time elapsed.
Use metrics to measure the quality of force laws

Fig. 4. The architecture of the simulation tool.

Our 2D simulation world is 900×700 in size, and contains a goal, obstacles and robots. Although we can use up to a maximum of 100 robots and 100 static obstacles with one static goal, we placed a compromise number of 40 robots and 90 obstacles in the environment when using the training module, because we were especially interested in determining whether the learned behavior would scale with the number of robots. The goal is always placed at a random position in the right side of the world, while the robots are initialized in the bottom left area. The obstacles are randomly distributed throughout the environment, but are kept 50 units away from the initial location of the robots and the goal to avoid proximity collisions. Each circular obstacle has a radius R_o of 10, and the square shaped goal is 20×20. When 90 obstacles are placed in the environment, roughly 4.5% of the environment is covered by the obstacles (similar to Balch and Hybinette [2000]). The desired separation between robots R is 50, and the maximum velocity V_{max} is 20. Figure 5 shows 40 robots navigating through randomly positioned obstacles. The larger circles are obstacles and the square to the right is the goal. Robots can sense other robots within a distance of $1.5R$, and can sense obstacles within a distance of $R_o + 20$. The goal can be sensed at any distance.

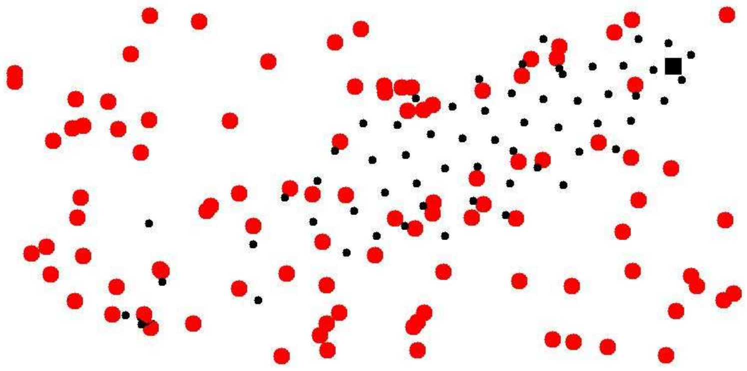


Fig. 5. 40 robots moving to the goal. The larger circles represent obstacles, while the square in the upper right represents the goal.

The environment generator creates task environments as in Figure 5 to test the force laws. The environment consists of robots, randomly positioned obstacles, and a goal. Each force law is tested on $n = 50$ different environment instances created by the environment generator. Each robot carries a copy of the force law and navigates towards the goal while avoiding obstacles. Robots are given a limited amount of time to accomplish the obstacle avoidance task and reach the goal while maintaining the formation. We refer to this as an evaluation run.

$$fitness_{ind} = \frac{\mathscr{R}_1 + \mathscr{R}_2 + \cdots + \mathscr{R}_n}{n} \tag{6}$$

The global observer (fitness function) evaluates the performance of the force law in an instance of the environment and assigns a fitness value, $\mathscr{R}_i$. Each evaluation run must be completed within a specific time interval, and the fitness assignment occurs at the end of the time interval which is also the end of an evaluation run. The final fitness, $fitness_{ind}$, of an individual is computed once n evaluation runs are completed.

Once the termination criteria of the EA is met, the EA outputs the optimal parameter setting for the force law that is being optimized. The termination criteria of our EA is 100 generations.

3.1. *Optimization using Evolutionary Algorithms*

Given generalized force laws, such as the Newtonian force law or Lennard-Jones (LJ), it is necessary to optimize the parameters to achieve the best performance. We achieve this task using an Evolutionary Algorithm (EA). EAs are optimization algorithms inspired by natural evolution. We mutate and recombine a population of candidate solutions (individuals) based on their performance in our environment. One of the major reasons for using this population-based stochastic algorithm is that it consistently generates individuals that have robust performance.

To optimize the force law parameters, we use the training module of our simulation tool. This training module allows the user to specify the type of force law, minimum and maximum parameter value bounds, the population size, the termination criteria, the mutation rate, and the crossover rate.

Every individual in the population is a vector of real-valued parameters, representing an instantiation of either the Newtonian or LJ force law (depending on the force law being optimized).

In addition to friction, the evolving parameters of the Newtonian force law are:

- G - gravitational constant of robot-robot interactions,
- p - power of the force law for robot-robot interactions,
- F_{max} - maximum force of robot-robot interactions,

and similar 3-tuples for obstacle/goal-robot interactions. The evolving parameters of the LJ force law are:

- ε - strength of the robot-robot interactions,
- c - non-negative attractive robot-robot parameter,
- d - non-negative repulsive robot-robot parameter,
- F_{max} - maximum force of robot-robot interactions,

and similar 4-tuples for obstacle/goal-robot interactions.

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.[a]. Each parameter has a $1/L$ probability of being mutated (L is the number of parameters in an individual). Mutation ensures that parameter values stay within accepted ranges.

Since we are using an EA that minimizes, the performance of an individual at each separate run i ($\mathscr{R}_i$ in equation 6) is measured as a weighted sum of penalties:

$$\mathscr{R}_i = w_1 \times P_{Collision} + w_2 \times P_{NoCohesion} + w_3 \times P_{NotReachGoal}$$

The weighted fitness function consists of three components: a penalty for collisions, a penalty for lack of cohesion, and a penalty for robots not reaching the goal.[b] Since there is no safety zone around the obstacles [Balch and Hybinette (2000)], a penalty is added to

[a]We used $\delta = 1.0$

[b]The results were robust with respect to reasonable values of the weights. We used $w_1 = 0.4, w_2 = 0.4, w_3 = 0.2$.

the score if the robots collide with obstacles. The cohesion penalty is derived from the fact that in a good (hexagonal) lattice, interior robots should have six local neighbors. A penalty occurs if a robot has more or less neighbors. If no robot reaches the goal within the time limit, a penalty occurs.

Figure 6 shows the evolved Newtonian robot-robot force law up to a distance of 75. A robot can sense another robot up to a distance of $1.5R$, where R is 50. The force is repulsive (< 0) when the distance between robots is less than 50, and it is attractive (> 0) when the distance is greater than 50. The evolved F_{max_r} takes effect when the distance between robots is less than 35.

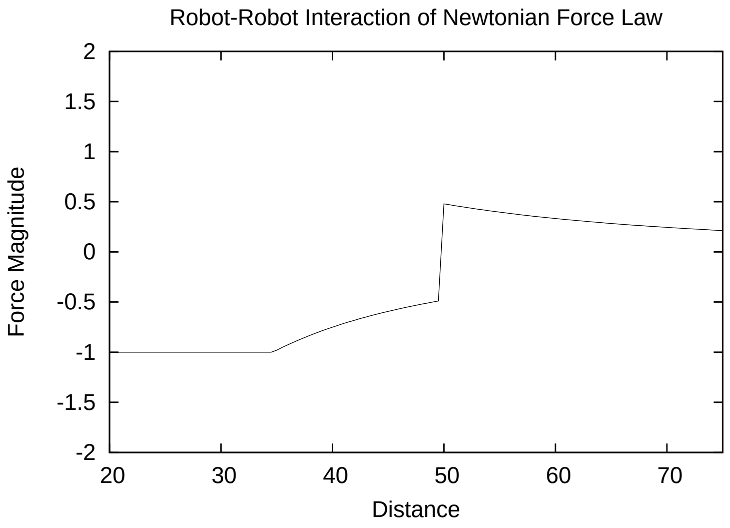


Fig. 6. Evolved Newtonian force law for robot-robot interactions.

Figure 7 shows the evolved LJ robot-robot force law. Force is repulsive when the distance between robots is less than 50, and it is attractive when the distance is greater than 50. The evolved F_{max_r} takes effect when the distance between robots is less than 45.

The permitted time interval for the robots to reach the goal from their initial position is set at 2000 simulation time steps. This accounts for approximately 47 seconds of clock time (we use a Linux-based dual processor Dell machine with Intel Xeon 1500MHz processors). The EA was run with 100 individuals per population and was allowed to terminate after 100 generations. It takes approximately five days for our EA to achieve a parameter set that provides the desired behavior regardless of the force law that is being optimized.

3.2. *Performance Metrics*

After optimization, the best force laws are evaluated with our performance module. The performance module consists of four metrics:

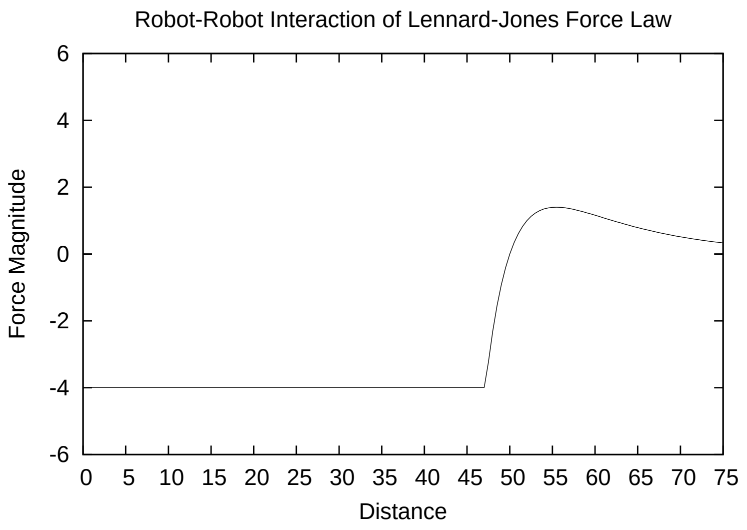


Fig. 7. Evolved LJ force law for robot-robot interactions.

- Collisions: the number of robots colliding with obstacles. We consider such robots to be damaged, but they can still move with the formation.
- Swarm connectivity: the maximum number of robots in the swarm that are connected via a communication path. Two robots are connected if their separation is $\leq 1.5R$.
- Reachability: the percentage of robots that reach the goal. A robot has reached the goal if it is within $4R$ distance of the goal.
- Time to goal: the amount of time taken by 80% of the robots to reach the goal. If the number of robots reaching the goal is less than 80%, we denote the time as '–'.

The importance of the collision, reachability, and time to goal metrics is obvious. We also consider connectivity, since this is an important metric for the quality of a swarm of robots acting as a sensor grid. The connectivity result we will provide is the minimum size of the largest connected swarm, as the swarm moves to the goal.[c] Although each metric provides useful information, a more complete picture arises by considering all.

3.3. *Simulation Results: Solid and Fluid Behaviors*

Both Newtonian and LJ force laws were evolved using our training module. The population size was 100 and the EA was run for 100 generations. We trained over scenarios with 40

[c]The connectivity metric is more informative than the prior cohesion metric, but is more expensive to compute. This is why it is only used after training.

robots and 90 obstacles, so that we could examine whether the evolved behaviors scaled well with an increasing number of robots.

To measure the performance of the optimized force laws, experiments were carried out with 20 to 100 robots (in increments of 20), and 20 to 100 obstacles (in increments of 20). Each experiment was averaged over 50 runs of different robot, goal and obstacle placements. A '–' entry indicates that the robots did not make it to the goal within the allotted time period.

Table 1. Summary of results for 100 obstacles, with 40 to 100 robots.

	Newtonian Force Law				LJ Force Law			
Number of Robots	40	60	80	100	40	60	80	100
Number of Collisions	0	3	5	7	0	1	2	4
Connectivity	27	60	80	100	23	37	53	67
Reachability%	29	1	0	0	97	98	98	98
Time to Goal	-	-	-	-	600	690	780	870

Since we trained with a large number of obstacles, the number of obstacles is not an important factor. However, the number of robots does affect performance, and we focus on this aspect here. Table 1 shows a summary of results with 100 obstacles. It is clear that collisions in both force laws are not a primary concern.

With the Newtonian force law, when there are 40 robots, 29% of the robots reach the goal. However, it is clear that this is achieved by fragmenting the formation into small parts. When there are more than 40 robots, none reach the goal (within the time period). Instead, the structure remains connected, but the strict rigidity of the structure prevents it from making good progress through the obstacle field. It is clear from these results that training with 40 robots does not yield a Newtonian force law that scales to a larger number of robots.

With the LJ force law, almost all of the robots make it to the goal, in all circumstances. The time to reach the goal increases slowly as the number of robots increases. Finally, swarm connectivity remains reasonably high, ranging from 58% to 67%. Interestingly, swarm connectivity increases as the number of robots increases, and is almost totally unaffected by the number of obstacles. In contrast with the Newtonian force law, the LJ force law (which is also trained with 40 robots) scales well with larger numbers of robots. This provides evidence that the LJ force law is a good model for the swarm behavior that we desire.

Observation of the system behavior shows that the LJ formation acts like a viscous fluid, rather than a solid. Although the formation is not rigid, it does tend to retain much of its structure. Deformations and rotations of portions of the fluid are temporary manifestations imposed by the obstacles. Hence, the added flexibility of this formation (over that achieved by the Newtonian force law) has a significant impact on behavior. The optimized LJ force law provides low collision rates, very high goal reachability rates within a reasonable period of time, and high swarm connectivity.

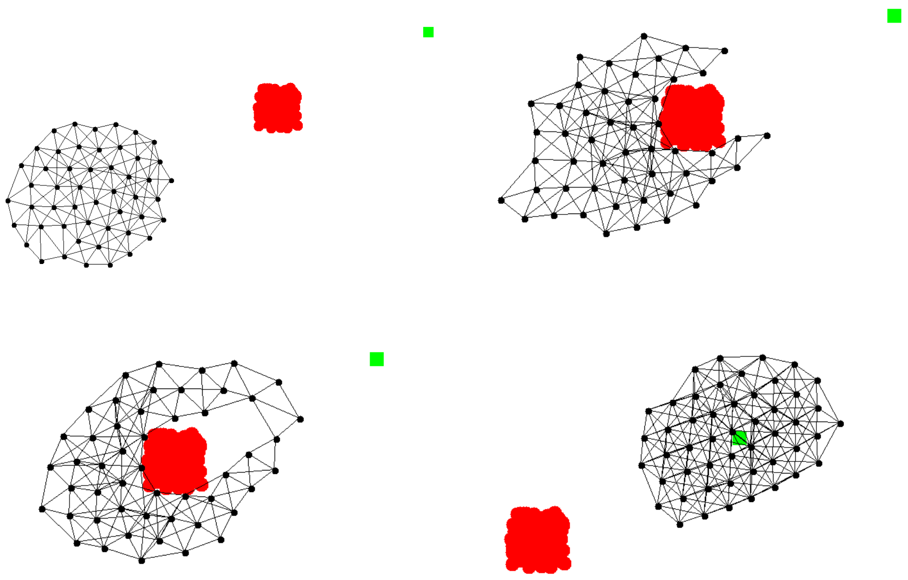


Fig. 8. Fifty robots navigating around a large obstacle toward a goal. Robots maintain full connectivity while avoiding the obstacle by acting as a viscous fluid, using the LJ force law.

Figure 8 shows a sequence of snapshots of 50 robots navigating around a large obstacle. Robots act as a viscous fluid while avoiding the obstacle. In the first snapshot, robots are in a fully connected sensor network and are navigating towards the goal, but the robots have not encountered the obstacle. The second snapshot shows the swarm starting to flow around the obstacle on two fronts while maintaining 100% connectivity. The third snapshot shows the robots on the two fronts merging back together. In the final snapshot, the robots are back in a cohesive formation when they have reached the goal. We observe that when the swarm reaches the obstacle, it navigates around the obstacle as a viscous fluid while maintaining 100% connectivity and provides 100% reachability. This fluid type property of the LJ force law is an emergent behavior of the swarm.

3.4. *Discussion and Elaboration*

To further analyze our system, we also collected data concerning the change in the connectivity and the percentage of robots reaching the goal, over time. The resulting graphs are far too numerous to present here, but we present representative examples. All graphs are averaged over 50 independent runs.

Figure 9 illustrates the change in connectivity of the swarm over time. Two sets of re-

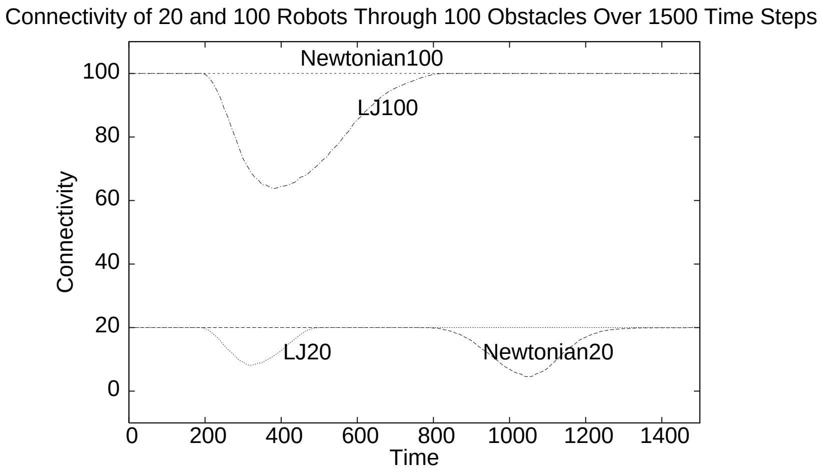


Fig. 9. Change in connectivity over 1500 time steps for 20 and 100 robots through 100 obstacles using Newtonian and LJ force laws

sults are presented in this graph. The curves at the top are for 100 robots moving through 100 obstacles. The robots controlled by the Newtonian force law remain fully connected (although, as we know from the prior results, this is because the formation has not succeeded in reaching the goal). However, the swarm connectivity for the LJ-controlled robots drops after 200 time steps, as the formation begins to move through the obstacle field. After 400 time steps, the formation connectivity increases as the robots reach the goal.

The curves at the bottom are for 20 robots moving through 100 obstacles. In this situation the Newtonian-controlled robots arrive at the goal, and the swarm connectivity drops after 800 time steps and then increases after roughly 1050 steps. Because the LJ-controlled formation moves much more quickly, the formation connectivity drops after 200 time steps and then increases after roughly 300 steps. It is interesting to note that the LJ-controlled swarm does not break apart quite as much as the Newtonian-controlled swarm.

Figure 10 shows how the number of robots reaching the goal changes with time. Again, two sets of results are presented, for 20 and 100 robots moving through 100 obstacles. The two left-most curves are for the LJ-controlled robots. Note that, regardless of the number of obstacles, robots start to arrive at the goal at roughly the same time (300 time steps). With 20 robots, they have all arrived at the goal by about 500 time steps. This indicates that all robots arrived at the goal within a 200 time step interval – a relatively narrow band in time. Increasing the number of robots to 100 increases the time interval to only 500 steps. The other two curves are for the Newtonian-controlled robots. With 20 robots, they start to reach the goal at 1000 time steps, and the interval is approximately 400 time steps. When there are 100 robots, none reach the goal within the allotted time period.

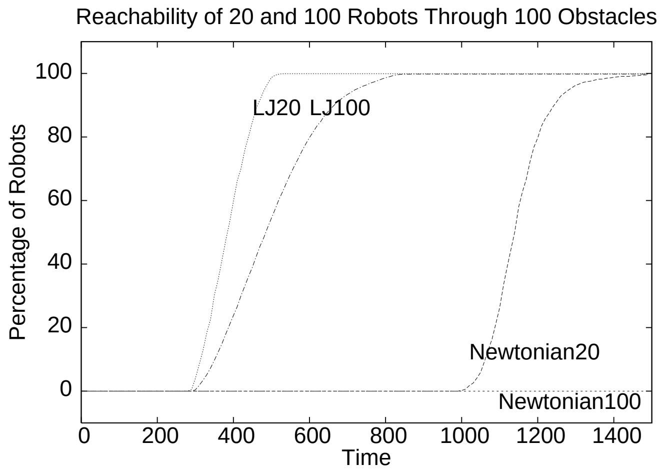


Fig. 10. The percentage of 20 and 100 robots reaching the goal through 100 obstacles over 1500 time steps using Newtonian and LJ force laws

3.5. *Summary*

We presented a novel extension to our *Physicomimetics* framework, with the use of a generalized Lennard-Jones force law. We then summarized how we used evolutionary algorithms to optimize the parameters of the force laws. These force laws were tested within the context of moving robotic swarm formations through obstacle fields to a goal.

In addition, we presented novel metrics of performance, namely, the number of robots that collide with obstacles, their connectivity, the number of robots that reach the goal, and the time to the goal. Although each metric provides useful information, a much better picture arises by considering all metrics. Our empirical analysis is methodical, ranging from 20 to 100 robots, and 20 to 100 obstacles.

Our results indicate that LJ-controlled robots have far superior performance to our more "classic" Newtonian-controlled robots. This is because the emergent behavior of the LJ-controlled swarm is to act as a viscous fluid, generally retaining good connectivity while allowing for the deformations necessary to smoothly flow through the obstacle field. Despite being trained with only 40 robots, the emergent behavior scales well to larger numbers of robots. In contrast, the Newtonian-controlled swarm produces more rigid structures that have much more difficulty maneuvering through the obstacles. Furthermore, performance drops dramatically when there are more than 40 robots.

4. Distributed Agent Evolution with Dynamic Adaptation to Local Unexpected Scenarios

We have shown how our *Physicomimetics* framework can be used to control formations of mobile robots that move towards a goal while avoiding obstacles (see Figure 5). An *offline* EA evolved an agent-level force law, such that robots 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)].

There are several difficulties with the previous approach of using an EA to evolve the behaviors of the agents and their local interactions. 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 versus homogeneity).

We propose a novel framework, called "Distributed Agent Evolution with Dynamic Adaptation to Local Unexpected Scenarios" (DAEDALUS) [Hettiarachchi *et al.* (2006)], for engineering multi-agent systems that can be used either offline or online. We will explore how DAEDALUS can be used to achieve global aggregate behavior, by examining two case studies pertaining to obstacle avoidance. In the first, obstacle density is tripled, far exceeding the norm in similar studies. In the second, obstructed perception is also modeled (the obstacles obstruct the perception of the robots). Both of these changes make the task far more difficult.

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". Agents that experience direct performance rewards have higher fitness. Other agents may not experience any direct reward, but may in fact have contributed to the agents that did receive direct reward. This "credit assignment" problem can be addressed in numerous ways, including the "bucket brigade" algorithm or the "profit sharing" algorithm [Grefenstette (1988)]. Assuming that a set A of agents has received some direct reward, both algorithms provide reward to the set B of agents that have interacted (and helped) those in A. Further trickle-back rewards are also given to those agents in set C that helped those in B, and so on. Agents that receive no rewards lose fitness. If fitness is low enough, agents stop moving or die.

Evolution occurs when individuals of the same species interact. Those agents with high fitness give their procedures to agents with lower fitness. Evolutionary recombination and mutation provide necessary perturbations to these procedures, providing increasing performance and the ability to respond to environmental changes. Different species may evolve different procedures, reflecting the different niches they fill in the environment.

4.1. *Online Approach with DAEDALUS*

Each robot of the swarm is an individual in a population that interacts with its neighbors. Each robot contains a *slightly mutated* copy of the optimized LJ force law rule set found with offline learning. This ensures that our robots are not completely homogeneous. We allowed this slight heterogeneity because when the environment changes, some mutations perform better than others. The robots that perform well in the environment will have higher fitness than the robots that perform poorly. When low fitness robots encounter high fitness robots, the low fitness robots ask for the high fitness robot's rules. Hence, better performing robots share their knowledge with their poorer performing neighbors.

When we apply DAEDALUS to obstacle avoidance, we focus on two aspects of our swarm: reducing obstacle-robot collisions and maintaining the cohesion of the swarm. Robots are penalized if they collide with obstacles and/or if they leave their neighbors behind. The second scenario arises when the robots are left behind in cul-de-sacs. This causes the cohesion of the formation to be reduced.

4.2. *Experimental Methodology of Online Adaptation*

Each robot of the swarm contains a slightly mutated copy of the optimized LJ force law rule set found with offline learning and all robots have the same fitness at the start. 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 online 2D world is 1650×950, which is larger than the offline world. In our changed environment, each obstacle has a radius of 30 compared to the offline obstacle radius of 10. More than 16% of the online environment is covered with the obstacles. Compared to the offline environment, the online environment triples the obstacle coverage. We also increase the maximum velocity of the robots to 30 units/sec, making the robots moves 1.5 times faster than in the offline environment. The LJ force law learned in offline mode is not sufficient for this more difficult environment, producing collisions with obstacles (due to the higher velocity), and robots that never reach the goal (due to the high percentage of obstacles). Figure 11 shows an example of the more difficult environment.

Robots that are left behind (due to obstacle cul-de-sacs) do not proceed to the next goal, but the robots that had collisions and made it to the goal are allowed to proceed to the next goal. We assume that damaged robots can be repaired once they reach a goal.

4.3. *DAEDALUS Results*

To measure the performance of the DAEDALUS approach, an experiment was carried out with 60 robots, 5 goals in the long corridor, and 90 obstacles in between each goal. The experiment was averaged over 50 runs of different robot, goal, and obstacle placements. Each robot is given equal initial fitness and "seeded" with a mutated copy of the optimized LJ force law learned in offline mode. If a robot collides with an obstacle, it's fitness is reduced. Whenever a robot encounters another robot with higher fitness, it takes the relevant parameters pertaining to the obstacle-robot interaction of the better performing robot.

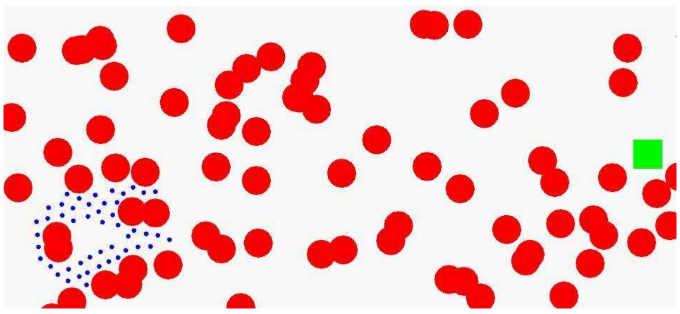


Fig. 11. 60 robots moving to the goal. The larger circles represent obstacles, while the square in the upper right represents the goal. The larger obstacles make this environment far more difficult for the robots to traverse.

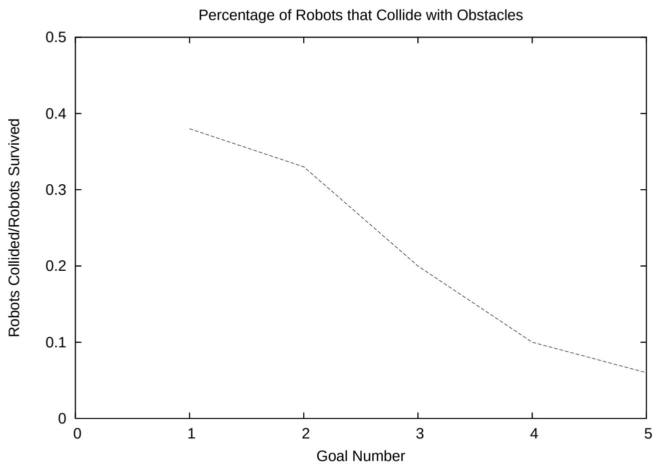


Fig. 12. The ratio of colliding robots versus the number of surviving robots, for 60 robots moving through 5 goals with 90 obstacles in between each goal.

Figure 12 shows the ratio of the number of robots that collided with obstacles versus the number of robots that survived to reach the goals. The graph indicates that after only 3 goals, the percentage of robots that collide with obstacles has dropped from about 38% to well under 10%. Inspection of the obstacle-robot parameters indicates that the repulsive component increased through the online process of mutation and the copying of superior force laws (this was confirmed via inspection of the mutated force laws).

This first experiment did not attempt to alleviate the situation where robots are left behind; in fact, only roughly 48% of the original 60 robots reach the final goal (see Figure 13, lower line). This is caused by the large number of cul-de-sacs produced by the large obstacle density. Our second experiment attempts to alleviate this problem by focusing on the robot-robot interactions. Our assumption was that the LJ force law needs to provide stronger cohesion, so that robots aren't left behind.

If robots are stuck behind in cul-de-sacs (i.e. they make no progress towards the goal) and they sense neighbors, they slightly mutate the robot-robot interaction parameters of their force laws. In a situation in which they do not sense the presence of neighbors and do not progress towards the goal, they rapidly mutate their robot-goal interaction causing a "panic behavior". These relatively large perturbations of the force law allow the robots to escape their motionless state.

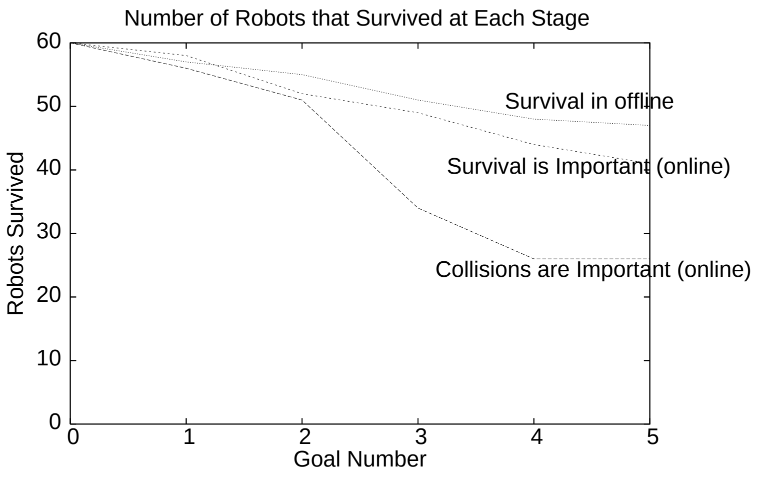


Fig. 13. A comparison of (a) the number of robots that survive when rules are learned using offline learning, (b) the number of robots that survive when using online learning (where the focus is on reducing collisions), and (c) the number of robots that survive when using online learning (and the focus in on survivability).

Figure 13 shows the results of this second experiment. In comparison with the first experiment (with survival rates of 43%), the survival rates have increased to 68%. As a control experiment, we ran our offline EA approach on this more difficult task. After five goals, the survival rate is about 78%. Recall that the offline results are obtained by running an EA with a population size of 100 for 100 generations, with each individual averaged over 50 random instantiations of the environment. As can be seen, the DAEDALUS approach provides results only somewhat inferior to the offline approach, in real time, while the robots are in the environment.

Although not shown in the graph, it is important to point out that the collision rates were not affected in the second experiment. Hence, we believe that it is quite feasible to combine both aspects in the future. Collision avoidance can be improved via mutation of the obstacle-robot interaction, while survival can be improved via mutation of the robot-robot interaction and robot-goal interaction.

5. Obstructed Perception

When a robot can not see another robot, due to the presence of obstacles, we call this "obstructed perception." When the robot's line of sight lies along an edge of an obstacle, the robots are capable of sensing each other. Surprisingly, this is not generally modeled in prior work in this area [Balch and Hybinette (2000)]. In the results given above, obstacles did not obstruct perception. The addition of obstructed perception makes the task far more difficult, especially as obstacle size increases [Hettiarachchi and Spears (2006)]. Figure 14 shows an example scenario of obstructed perception. The larger circle represents an obstacle, and *A* and *B* are robots. We define *minD* to be the minimum distance from the center of the obstacle to the line of sight of robot *A* and robot *B*, and *r* is the radius of an obstacle. If $r > minD$, then robot *A* and robot *B* have their perception obstructed.

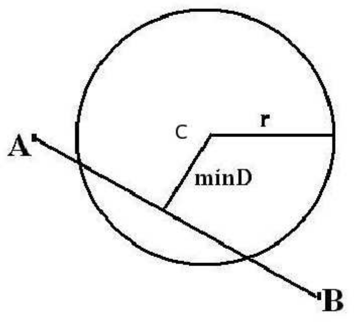


Fig. 14. Sensing capability of two robots (A,B) is obstructed by a large obstacle (C).

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

$$minD = \sqrt{(((1-q)\times X_a + q\times X_b) - X_c)^2 + (((1-q)\times Y_a + q\times Y_b) - Y_c)^2} \tag{7}$$

where (X_a, Y_a) and (X_b, Y_b) are the x,y positions of robots *A* and *B*, (X_c, Y_c) is the position of the center of an obstacle, and *q* is the minimum function that is defined by:

$$\frac{((X_c - X_a)\times(X_b - X_a) + (Y_c - Y_a)\times(Y_b - Y_a))}{\left((X_b - X_a)^2 + (Y_b - Y_a)^2\right)} \tag{8}$$

5.1. *Results with Obstructed Perception*

We compared DAEDALUS to three control studies. In the first control study, we train the robots 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 robots 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 robots 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 to large obstacles.

Figure 15 shows the results. The *y*-axis gives the number of robots that survived to reach the goal at each stage for the four different experiments. The top performance curve is for the first control study. Note that learning with small obstacles in offline mode is not hard, and the robots perform very well in the online environment. This is due to the fact that the small obstacles make the environment less dense providing the robots sufficient space to navigate. Out of 60 initial robots released in the online environment, 93.3% survived to reach the last goal. With such small obstacles (which is the maximum density examined in the related literature), obstructed perception is not an important issue.

As presented earlier, robots that learned without obstructed perception on larger obstacles had a reasonably high survival rate (78%). The bottom (dashed) performance curve shows the effect of obstructed perception (the second control study). Learning with large obstacles in offline mode with obstructed perception is very difficult, and the test results show that out of 60 robots released initially into the online environment only 35% (21 robots) survived to reach the last goal. This is due to the fact that the environments with larger obstacles create large numbers of cul-de-sacs that obstruct perception.

The third control study, where offline training occurs with small obstacles and testing occurs with large obstacles, is surprisingly good (see “NO DAEDALUS (small-large)”). Despite an initial drop in performance, performance at the fifth goal is quite acceptable (out of the initial 60 robots, 40% (24 robots) survived to reach the final goal). This is a 5% improvement over the robots that were trained on larger obstacles. These results run counter to accepted wisdom, which states that it is best to train on the hardest environments that you will encounter. In fact, this example demonstrates that training on simpler problems and applying the knowledge gained to harder problems can potentially provide superior results. Why is this so? As with developmental psychology, one does not train children on hard problems immediately, instead, we train them on easier problems first, in the hopes that they will learn the “basics” (which are important building blocks for solving other, more difficult, problems) more quickly.

If we extend the developmental psychology analogy further, we note that we encourage children to experiment and modify their behavior, based on changes in the environment. Furthermore, they share the lessons learned. This is precisely what the DAEDALUS system does. The final performance curve in Figure 15 shows the results. With an initial 60 robots, 58.3% or 35 robots survived to reach the last goal. This is a 23.3% improvement over the robots that learned in an environment with the larger obstacles, and a 18.3% improvement over the robots that learned with small obstacles and tested with the larger obstacles without DAEDALUS. These preliminary results are very promising. Although encouraging the robots (or children) to explore and experiment does provide an early drop-off in performance (compared to the “NO DAEDALUS (large-large)” curve), the results after four goals are superior. This is a classic example of “exploration” versus “exploitation”. Pure exploitation of learned knowledge is good up to a point, but will eventually

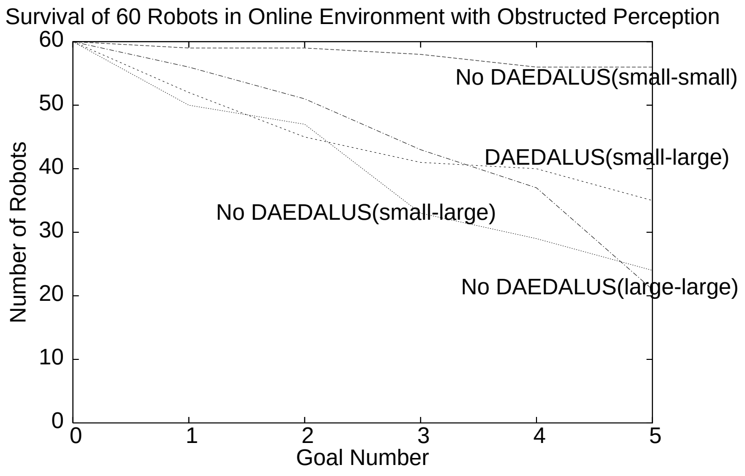


Fig. 15. Four different experiments of number of robots surviving - all robots are trained with obstructed perception and tested with and without DAEDALUS. The results are averaged over 100 independent runs.

fail as the problems become more difficult. Exploration provides the key to adapt to these changing environments. DAEDALUS provides just this form of exploration.

5.2. *Homogeneous DAEDALUS Results*

For the DAEDALUS performance curve given above, all robots had the same mutation rate, which was 5%. Hence, each robot had the same rate of exploration. Although the rules for each robot may differ, their mutations rates are identical, and we refer to this system as "Homogeneous DAEDALUS". However, there are numerous problems with this approach. First, the results may depend quite heavily on choosing the correct mutation rate. How is this mutation rate to be chosen? Second, the best mutation rate may also depend on the environment, and should potentially change as the environment changes. How is this to be accomplished?

Since the mutation rate may have a major effect on performance, we decided to explore this effect by conducting several experiments with different mutation rates. Figure 16 shows five independent experiments of Homogeneous DAEDALUS. Five different mutation rates were used: 1%, 3%, 5%, 7%, and 9%. The results are quite striking. Of the five different mutation rates, only 5% and 7% did well (with about 35 robots surviving to the last goal). Recall that the DAEDALUS performance curve shown in Figure 15 resulted from an arbitrarily chosen mutation rate of 5%. As it turns out, we were extremely fortunate in our design decision. For example, with mutation rates of 1%, 3%, and 9%, at most 20 robots survive to reach the final goal. The performance curve for the 9% mutation rate is especially interesting. Although promising at first, it appears as if the mutation rate is so high that it eventually causes an extremely deleterious mutation to appear. Mutation rates of 1% and 3% are too low to cope with the changed environment.

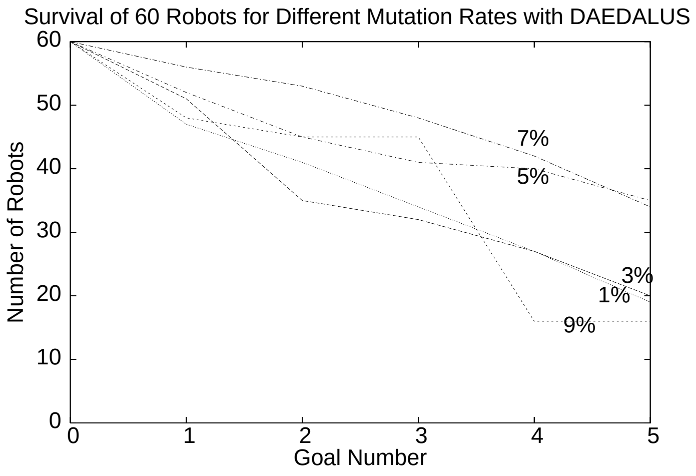


Fig. 16. Five different mutation experiments of robots surviving - all robots are trained with obstructed perception and tested with DAEDALUS. The results are averaged over 100 independent runs.

5.3. *Heterogeneous DAEDALUS Results*

In an attempt to address the problem of choosing the correct mutation rate, we divided the robots into five groups of equal size. Each group of 12 robots was assigned a mutation rate of 1%, 3%, 5%, 7%, and 9%, respectively. This mimics the behavior of children that have different “comfort zones” in their rate of exploration. Since different robots have different mutation rates, we refer to this system as “Heterogeneous DAEDALUS”. Figure 17 shows the results, in comparison with the three control studies shown in Figure 15. The label “Het.DAEDALUS(small-large)” shows the survivability of robots with pre-assigned mutation rates. Out of the initial 60 robots, 27 or 45% robots survived to reach the final goal. Although this is higher than our second and third control studies, it did not produce results as good as the results achieved with Homogeneous DAEDALUS using a 5% mutation rate (as shown in Figure 16). In fact, the result at the final goal is essentially identical to the average of the five performance curves shown in Figure 16.

5.4. *Extended Heterogeneous DAEDALUS Results*

In an attempt to improve performance, we again borrowed from the analogy of a “swarm” of children learning some task. Not only do they share useful information as to the rules they might use, but they also share meta-information as to the level of exploration that is actually safe! Very bold children might encourage their more timid comrades to explore more than they would initially. On the other hand, if a very bold child has an accident, the rest of the children will become more timid. In “Extended Heterogeneous DAEDALUS”, five groups of children are again initialized with mutation rates of 1%, 3%, 5%, 7%, and 9%. However, in this situation, if a robot receives the rules from a neighbor (which, again, occurs if that robot is in trouble), it also receives the neighbor’s mutation rate. In this imple-

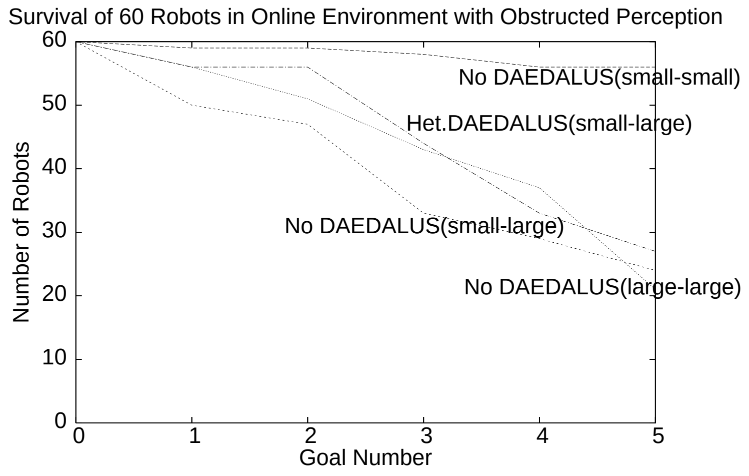


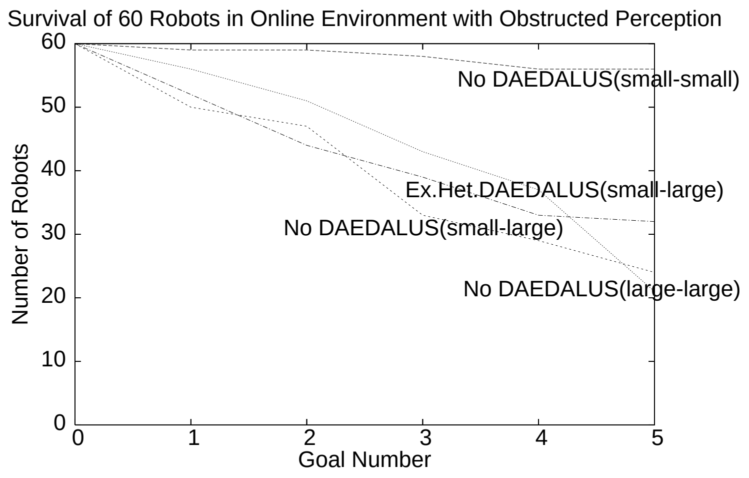
Fig. 17. The number of robots surviving with pre-assigned mutation rates. Mutation rates are not exchanged - all robots are trained with obstructed perception and tested with or without DAEDALUS. The results are averaged over 100 independent runs.

mentation, children in trouble not only change their rules, but their mutation rate. Figure 18 shows the results of this study. The curve labeled with "Ex.Het.DAEDALUS(small-large)" refers to the survivability of robots with pre-assigned mutation rates that also allows the robots to receive a neighbor's mutation rate, if the robot receives the neighbor's rules. The behavior is quite good. On average, 32 robots survive to reach the final goal, which is very close to the optimum value of 35 found by the best Homogeneous DAEDALUS experiment.

5.5. *Effect of Mutation in Swarm Learning*

We explored the effect of heterogeneous swarms in an online environment and compared our results with the offline homogeneous swarms. We maintained the diversity in our heterogeneous swarm by allowing robots to exchange their predefined mutation rates. The robots learned to avoid cul-de-sacs in the online environment and maintain the diversity of the population. Table 2 shows the mutation rates of robots that survive to reach a goal, averaged over 20 runs.

At the beginning, there are five groups of robots. They are initialized with mutation rates of 1%, 3%, 5%, 7%, and 9%. The robots with 1% and 3% mutation rates had a more difficult time surviving compared to the robots with other three mutation rates. Thirty seven robots survived to reach the fifth goal, and clearly the 5% and the 7% mutation rates performed better than the other three mutation rates. With 1% mutation, seven robots did not reach the fifth goal, and with 9% mutation, five robots did not reach the fifth goal. Notice that there are still robots with all five mutation rates surviving in the environment. This still maintains the diversity of the swarm.



]

Fig. 18. The number of robots surviving with pre-assigned mutation rates. Mutation rates are exchanged - all robots are trained with obstructed perception and tested with or without DAEDALUS. The results are averaged over 100 independent runs.

	Mutation Rate				
robots 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 robots that survive to reach a goal and their mutation rates.

5.6. *Summary*

Traditional approaches to designing multi-agent systems are offline, and assume the presence of a global observer. However, this approach will not work in real-time online systems. We presented a novel approach to solving this problem, called DAEDALUS, where we showed how concepts from population genetics could be used with swarms of agents to provide fast online adaptive learning in changing environments.

We addressed the important issue of “obstructed perception” in learning behaviors for swarms of robots that must avoid obstacles while reaching a goal. This issue has been largely absent from the literature. Our obstacle density is also three times higher than the norm, making obstacle avoidance a far more difficult task. Since obstructed perception makes the task far more difficult, DAEDALUS had to be extended. Our first extension was to allow different robots to have different rates of exploration, which affects the rate at which they change their behavioral rules. The second extension allows robots to also share their rates of mutation, allowing robots to find the right balance between exploration and exploitation. Results of the extended system are almost as good as the best results we were able to achieve when the exploration rates were controlled by hand. Our framework allows swarms of robots to not only learn and share behavioral rules in changing environments (in real time), but also to learn the proper amount of behavioral exploration that is appropriate (see [Hettiarachchi (2007)] for full results).

6. Related Work

In the specific context of obstacle avoidance, the most relevant papers are [Balch and Arkin (1998)], [Balch and Hybinette (2000)] and [Fredslund and Matarić (2002)]. Balch and Arkin [1998] examines the situation of four robots moving in formation through an obstacle field with 2% coverage. In Balch and Hybinette [2000], they extend this to an obstacle field of 5% coverage, and also investigate the behavior of 32 robots moving around one medium size obstacle. Fredslund and Matarić [2002] examine a maximum of eight robots moving around two wall obstacles. To the best of our knowledge, we are the first to systematically examine larger numbers of robots and obstacles.

The work done in OH́ara *et al*. [2005] uses an embedded network distributed throughout the environment to approximate the path-planning space and uses 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.

In the specific context of adaptive learning of multi-agent systems, some of the relevant papers are [Watson *et al*. (2002)] , [Crawford and Veloso (2005)], [Kira and Schultz (2006)], and [Goldman and Zilberstein (2003)]. The "Embodied Evolution" concept presented in Watson *et al*. [2002] is applied to eight light seeking physical robots that use a neural-network controller. This work is conceptually similar to ours and was developed independently. Their robots broadcast genetic information over their local-range communication channel, and the other robots who receive this genetic information are allowed to overwrite their own genetic information. Our robots do not broadcast their genetic information, rather they seek genetic information from their neighbors, if there are neighbors, otherwise they alter their own genetic makeup through mutation. They accept only the genetic information that is required to improve their current situation, i.e. a robot that is stuck behind a cul-de-sac accepts its neighbor's robot-obstacle genetic information to change its state from stuck to moving. DAEDALUS minimizes the communication overhead related to broadcasting and requirements for complex communication protocols.

Crawford and Veloso [2005] examines the Multi-Agent Meeting Scheduling problem where distributed software agents negotiate meeting times on behalf of their users. The agents learn online which strategies to use when negotiating with different agents by observing its own rewards as opposed to trying to model the other agents. In this work, the agents do not use an evolutionary approach for agent-agent negotiation, and do not share their control structures to maximize their utility, as in our work.

The work done in Kira and Schultz [2006] uses rule-based robots to continually improve their control system by applying a learning algorithm on an internal simulation of its environment and to update this simulation to reflect major changes within the environment. Goldman and Zilberstein [2003] presents a distributed decision-theoretic solution to a multiple decision making multi-agent system that shares a common set of objectives. This theoretical formal model based on Markov Decision Processes enables the study of the trade-off between the cost of information and the value of the information acquired in the communication process and its influence on the joint utility of the agents.

7. Summary and Conclusion

In this paper examined the performance of *Physicomimetics* on an obstacle avoidance task, comparing our standard "Newtonian" force law with the Lennard-Jones (LJ) force law. Our results indicate that the LJ-controlled robots have far superior performance to our Newtonian-controlled robots. This is because the emergent behavior of the LJ-controlled swarm is to act as a viscous fluid, generally retaining good connectivity while allowing for the deformations necessary to smoothly flow through the obstacle field. Despite being trained with only 40 robots, the emergent behavior scales well to larger numbers of robots. In contrast, the Newtonian-controlled swarm produces more rigid structures that have more difficulty maneuvering through the obstacles. Furthermore, performance drops dramatically when there are more than 40 robots.

In addition, we presented novel metrics of performance, namely, the number of robots that collide with obstacles, their connectivity, the number of robots that reach the goal, and the time taken by at least 80% of the robots to reach the goal. Although each metric provides useful information, a much better picture arises by considering all metrics. Our empirical analysis is methodical, ranging from 20 to 100 robots, and ranging from 20 to 100 obstacles.

Finally, we proposed a novel framework, called "Distributed Agent Evolution with Dynamic Adaptation to Local Unexpected Scenarios" (DAEDALUS), for engineering multi-agent systems that can be used either offline or online. We showed how concepts from population genetics can be used with swarms of agents to provide fast online adaptive learning in changing environments using DAEDALUS. Our framework allows swarms of robots to not only learn and share behavioral rules in changing environments (in real time), but also to learn the proper amount of behavioral exploration that is appropriate. We addressed the important issue of "obstructed perception" in learning behaviors for swarms of robots that must avoid obstacles while reaching a goal. This issue has been largely absent from the literature. Our obstacle density is also three times higher than the norm, making obstacle avoidance a far more difficult task.

Currently we are building an Obstacle Avoidance Module (OAM) for our outdoor robots, so that we can test the utility of the Lennard-Jones potential and DAEDALUS in real world situations. Initial results have been promising (see Figure 19).

The future directions of our work will focus on improving and extending our contributions as well as applying them to provide practical solutions to complex problems. A

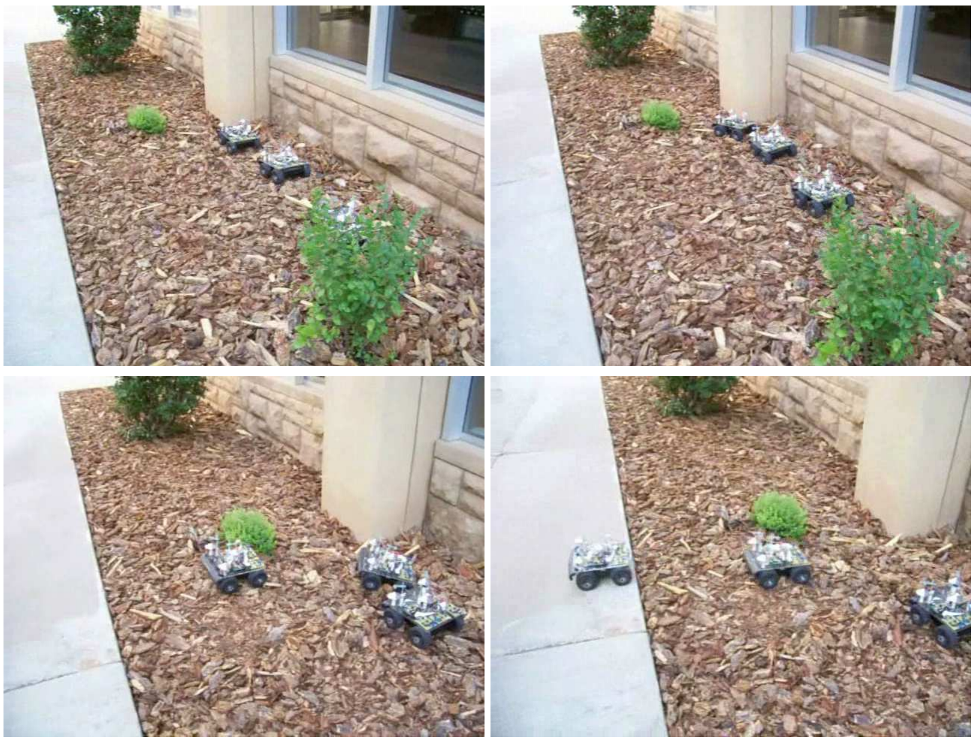


Fig. 19. Three outdoor robots with *Physicomimetics* and an obstacle avoidance module.

significant portion of this work is dedicated to exploring the issue related to partial observation. Still, there are significant issues that arise with respect to“wall following methods” and “local minimum trap” problems. These issues are not adequately addressed in this paper. We have observed “local minimum trap” problems in our work, but we did not make attempts to address this issue in detail. We intend to introduce a hybrid liquid and gas model combined with our DAEDALUS approach as a solution. The robots will switch to a gas model as presented in [Kerr *et al*. (2005)] to avoid the “local minimum trap”. Once the robots have escaped they can continue using the previous force law. The performance metrics we define provide future researchers with meaningful benchmarks of swarm behavior. A possible extension to these metrics could provide the distribution of sub-swarms and the number of robots in each sub-swarm.

The results of our heterogeneous swarms are promising, but we believe that robot behavior can be further improved by different mutation techniques. We intend to explore other approaches to develop more robust adaptive algorithms for online learning. We believe that we can accelerate the learning of the mutation rates. For example, currently, when a robot is in trouble, it receives the rules and mutation rate of a neighbor that is not in trouble. But this same neighbor could also query the robot in trouble to find out its mutation rate. Then the neighbor could spread this information further, to inform other robots that this particular mutation rate might be problematic. Also, another possible avenue for improving the performance of our DAEDALUS approach lies within reward sharing (i.e. credit assignment) techniques. 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 learning of heterogeneous swarms.

Acknowledgements

We thank the Joint Ground Robotics Enterprise for funding the obstacle avoidance portion of this work. We also thank Professor Tom Bailey for some mathematical corrections.

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.*

Crawford, E. and Veloso, M. (2005). Learning to Select Negotiation Strategies in Multi-Agent Meeting Scheduling. *In the working notes of the Multiagent Learning Workshop, AAAI.*

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.

Goldman, C. V. and Zilberstein, S. (2003). Optimizing Information Exchange in Cooperative Multiagent Systems. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems,* 137–144.

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. (2007). Distributed Evolution for Swarm Robotics. *Ph.D. Thesis, Department of Computer Science, University of Wyoming..*

Howard, A.,Matarić, M., and Sukhatme G. (2002). Mobile Sensor Network Deployment Using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. *Proceedings of the Sixth International Symposium on Distributed Autonomous Robotics Systems,* 299–308.

Kerr, W., Spears, D., Spears, W. and Thayer D. (2005): Two formal gas models for multiagent sweeping and obstacle avoidance. *Lecture Notes in Artificial Intelligence, Springer,* **3228**: 111–130.

Khatib, O. (1986). Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *International Journal of Robotics Research. MIT Press,* 90–98.

Kira, Z., and Schultz, A. C. (2006) Continuous and Embedded Learning for Multi-Agent Systems. *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 3184–3190

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.

Reif, J. and Wang, H. (1998): Social potential fields: A Distributed Behavioral Control for Autonomous Robots. *Workshop on the Algorithmic Foundations of Robotics,* 431–459.

Schoenwald, D., Feddema, J. and Oppel, F. (2001). Decentralized Control of a Collective of Autonomous Robotic Vehicles. *American Control Conference,* 2087–2092.

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., Heil, R., Spears, D. and Zarzhitsky D. (2004). *Physicomimetics* for Mobile Robot Formations. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems,* 1528–1529.

Spears, W., Spears, D., Heil, R., Kerr,W. and Hettiarachchi, S. (2005). An overview of *Physicomimetics*. it In Şahin, E. and Spears, W., Eds.:Lecture Notes in Computer Science State-of-the-Art Series, Springer, **3342**: 84–97.

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

Spears, W., Hamann, J., Maxim, P., Kunkel, T., Heil, R., Zarzhitsky, D., Spears, D. and Karlsson, C. (2006). Where are you? *In Şahin, E., Spears, W., eds.: Swarm Robotics, Lecture Notes in Computer Science, Springer,* **4433**: 129–143.

Vail, D. and Veloso M. (2003). Multi-robot Dynamic Role Assignment and Coordination Through Shared Potential Fields. *Multi-Robot Systems, Kluwer.*

Watson, R., Ficici, S. and Pollack, J. (2002). Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots. *Robotics and Autonomous Systems, Elsevier,* **39**: 1–18.

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.

About the authors

Suranga Hettiarachchi received his Ph.D. in Computer Science from the University of Wyoming in 2007. He is currently an assistant professor at Indiana University Southeast and is also the founder of the swarm robotics laboratory there. His research interests include multi-agent systems, *Physicomimetics*, distributed robotics, evolutionary learning, search algorithms, adaptive learning, and swarm intelligence. His publications have appeared in refereed international journals and conferences. He can be contacted at *suhettia@ius.edu*.

William M. Spears received a Ph.D. in Computer Science from George Mason University in 1998. He has an international reputation for his expertise in evolutionary computing and has a published book on the topic. He has co-edited books on swarm robotics and evolutionary computation. His current research includes distributed robotics, the epidemiology of virus spread, evolutionary algorithms, complex adaptive systems, and learning and adaptation. He was co-founder of the University of Wyoming Distributed Robotics Laboratory and has approximately 75 publications. He is the CEO of Swarmotics, LLC. He can be contacted at *wspears@swarmotics.com*.