A Generalized Graph-Based Method for Engineering Swarm Solutions to Multiagent Problems*

R. Paul Wiegand[1], Mitchell A. Potter[1], Donald A. Sofge[1], and William M. Spears[2]

[1] U.S. Naval Research Lab
{wiegand, mpotter, sofge}@aic.nrl.navy.mil
[2] University of Wyoming
wspears@cs.uwyo.edu

Abstract. We present two key components of a principled method for constructing modular, heterogeneous swarms. First, we generalize a well-known technique for representing swarm behaviors to extend the power of multiagent systems by specializing agents and their interactions. Second, a novel graph-based method is introduced for designing swarm-based behaviors for multiagent teams. This method includes engineer-provided knowledge through explicit design decisions pertaining to specialization, heterogeneity, and modularity. We show the representational power of our generalized representation can be used to evolve a solution to a challenging multiagent resource protection problem. We also construct a modular design by hand, resulting in a scalable and intuitive heterogeneous solution for the resource protection problem.

1 Introduction

Natural examples of emergent complexity from collections of simple components have led to the development of a number of methods that provide *swarm intelligence* — collective capabilities from simple autonomous agents [1]. Application of swarm methods to discrete and real-valued optimization problems include ant colony optimization [2] and particle swarm optimization [3] respectively, while other swarm methods have been applied to the area of collective robotics [4]. Designing swarms in simple situations is primarily a matter of replicating agents with the same behaviors, but more challenging problems require varying degrees of heterogeneity, where agents may *share* key behaviors and may also be capable of specialization. However, few swarm methods address issues of heterogeneity and modularity.

Historically, problems in Artificial Intelligence have been approached using methods that involve representing and incorporating domain knowledge. Unfortunately, such methods are difficult to implement, due to the amount of human engineering required. This is especially true for multiagent problems, where the number of interactions between agents becomes prohibitive. In response, swarm-based solutions to multiagent problems have been knowledge-poor. This raises other issues, especially with respect to scalability and intuition. What is missing is a principled and practical method for finding a middle ground: incorporating *some* human knowledge into the system, while providing as much representational flexibility as possible.

* This work was performed under Office of Naval Research Work Orders N0001406WX20006 and N0001406WX30002.

We present a method for designing swarm-based behaviors for multiagent teams that achieves this objective. While our method is general, this paper will focus on one particular swarm control paradigm, *physicomimetics*. Specifically, we will illustrate how physicomimetics can be generalized to include heterogeneity explicitly as part of the swarm design process, and will introduce a graph-based method to design heterogeneous, modular swarms. Our method allows engineers to embed knowledge about the domain into the system to constrain agent interactions for improved scalability, as well as to maintain intuition about the system's operation.

The paper first presents background information on physicomimetics, then describes our generalizations to this framework. An example of how this can be used to develop heterogeneous solutions is given. We follow by describing our graph-based design method and use it to construct a heterogeneous, modular solution by hand. The result is a very scalable solution that was intuitively designed and easily understood. We finish by discussing how our work relates to other swarm engineering methods, then provide some concluding remarks, including intended future endeavors.

2 Physicomimetics

Physicomimetics provides a framework for the control of multiple agents [5]. Agents are treated as point-mass (m) particles. Each particle has a position, $\mathbf{x}$, and velocity, $\mathbf{v}$. We use a discrete time simulation, with time-step Δt. At each time step, the particle is repositioned based on the velocity and the size of the step, $\Delta\mathbf{x} = \mathbf{v}\Delta t$. The change in velocity of the particles is determined by the artificial forces operating on the particles, $\Delta\mathbf{v} = \mathbf{F}\Delta t/m$, where $\mathbf{F}$ is the aggregate force on the particle as a result of interactions with other particles and the environment. Each particle also has a coefficient of friction, $c_f \in [0, 1]$. Velocity in the next step becomes $(\mathbf{v} + \Delta\mathbf{v})c_f$, stabilizing the system [6,7].

There are two constraints: the magnitude of the force cannot exceed F_{max} and the magnitude of the velocity cannot exceed V_{max}. These restrict acceleration and velocity of particles in the model. Also, since there is an emphasis on *local* interactions, there are further restrictions on the range of effect particles have on other particles.

The simplicity of this framework creates a number of benefits. First, a variety of force laws can be employed to different effect. Moreover, the parameters of the above model, coupled with the force law parameters, provide engineers with mechanisms to adjust the behaviors of agents. Finally, since physicomimetics is based on physics, practical analyses are possible using traditional physics techniques such as force balance equations, conservation of energy and potential energy [6,8].

A slight variation of the well-known Newtonian force law will be used in this paper. The range of effect of the force is C, while R is the desired *range of separation* between agents. The gravitational constant is G, and there are two parameterized exponents: one for distance, d, and one for mass, a. The distance between two particles i and j is r_{ij}. The magnitude of the force between particles i and j is computed as follows.

$$F_{ij} = \begin{cases} -G\frac{(m_i m_j)^a}{r_{ij}^d} & \text{if } r_{ij} \in [0, R) \\ G\frac{(m_i m_j)^a}{r_{ij}^d} & \text{if } r_{ij} \in [R, C] \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

This force law repels particles closer than R and attracts particles past that distance but within the range of effect. The gradient of the force can be controlled using d, and a can raise or lower the importance of mass on the force. In total, there are two parameters associated with each particle (m and c_f) and five parameters associated with their interactions (G, C, R, a, and d). Distance variable r_{ij} is an observed phenomenon.

3 Generalizing Physicomimetics

3.1 Differentiating Particle Types and Their Interactions

In heterogeneous multiagent systems, different types of agents will have different behavioral profiles. When heterogeneity is necessary, the first step is to explicitly consider different particle types during the swarm design process — each type having its own mass and coefficient of friction. Differentiating particle types provides some degree of specialized behaviors. For example, we can generate a variety of ring formations of arbitrary radii by creating two different particle types: one with a relatively small mass and one with a relatively large mass. However, only a limited subset of ring behaviors are possible when all particles are homogeneous.

In addition to differentiating particles, interactive forces between the types of particles can vary. When heterogeneity is important, the second step a swarm engineer should consider is specializing the different interactions between those types. With the combination of different types of interactions and different particle types, a wide range of complex heterogeneous behaviors are now possible. Moreover, by controlling how many particle types there are, and how many agents there are of each particle type, engineers can explicitly control the *level of heterogeneity* in cooperative teams.

In the simplest cases, the same force law is imposed for all interactions, but the parameters differ. For example, Spears *et al.* [6] showed that, while one can generate hexagonal lattice formations using traditional Newtonian physics, to produce square lattices one must differentiate particles[1] and vary the parameters of their interactions.

Force interactions between different particle types may also be asymmetric. That is, particle type A may affect particle type B differently than B affects A. This idea was leveraged by the online evolutionary learning system applied by [9,10] to an obstacle avoidance problem. In this case, particles representing agents reacted to each other differently than those representing the goal or the obstacles, yet the particles representing the goal and obstacles remained fixed.

More generally, the underlying force law itself can vary for different interactions. There is a physical metaphor for this — particles in the natural world affect one another via a variety of forces and one often dominates. An example of where this might be useful is the game capture the flag. Those agents retrieving the opponent flag might be better off using a force law that takes advantage of fluid-like effects for movement and obstacle avoidance, while those protecting the home flag may be better off using something more appropriate for strong structural formations [10].

[1] In the referenced work, they used the artificial label "spin" to differentiate particles.

3.2 An Example Problem

To begin exploring the advantages and limitations of our generalized physicomimetic framework, we introduce a simple resource protection problem. A centrally-located, immobile resource is encircled by a defense perimeter. Nine protector agents are deployed from the vicinity of the resource. Two slightly faster intruder agents appear in random locations just outside the perimeter and begin attack runs at the resource, attempting to avoid protectors during the run. If an intruder is destroyed, hits the resource, or is chased out of the perimeter, it is removed from the simulation, and a new intruder will begin a new run from just outside the perimeter after a short random waiting period.

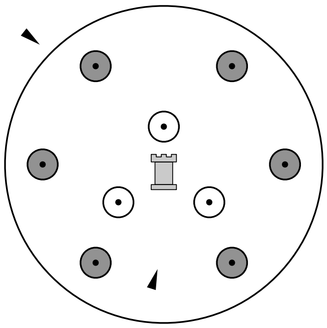

Fig. 1. Example resource protection problem. Castle marks resource to be protected, outer circle marks defense perimeter, gray and white circles with dots indicate protectors (two types), and the triangles indicate intruders (one type).

The problem is multiobjective. Ultimately, we want to reduce the extent of incursions into the defense perimeter, but also we want to avoid damage to the resource. Thus we define two objectives: the average per-step incursion distance of intruders into the perimeter and the ratio of resource hits taken over the total number of intruder runs at the resource. While these two objectives overlap a great deal, they are not the same — particularly when there are multiple intruders as is the case here.

As a result of its multiobjective nature, the resource protection problem is a good one for exploring questions about heterogeneity. By changing aspects of the problem, such as the relative importance of the two objectives, the number of intruders, or the types of possible intruder behaviors, we can begin to address questions about how heterogeneous teams of protectors can help, and what level of heterogeneity is useful in what circumstances. Specifically, we will consider solutions that allow for two different types of protectors (6 of one and 3 of the other) defending against a single type of intruder. The intuition here is to allow the system to deal with each objective separately byproviding it with different protective mechanisms for each objective. For example, it might be useful to have one set of protectors on the frontier chasing away intruders as soon as they enter the perimeter, while also having a few protectors back by the resource to prevent last-minute strikes.

3.3 Optimizing a General Physicomimetic Solution

Solutions to the resource protection problem can be represented using generalized physicomimetics; intruders and protectors use this model in all cases in this paper. Additionally, since their behaviors will be significantly influenced by the resource itself, it is also useful to model this as a separate particle type.

If we allow all possible instantiations, the parameter space is quite large, but we can reduce it somewhat. Since the central resource does not move, we need not worry about its coefficient of friction or interactions from other types of particles. Additionally, we limit the intruders to a single pre-defined strategy and focus on optimizing a solution for the protectors. Still, protector behaviors require eight interactions (one from each of the four types affecting each of the two protector types) and a total of 46 parameters (8 interactions with 5 parameters each + 4 for the mass of each type + 2 for the c_f of the protectors). If we were to add another protector type, there would be 6 more interactions and 32 more parameters. Indeed, the number of parameters scales quadratically with the number of particle types.

In spite of these simplifications, the size and complexity of the parameter space make it intractable for us to solve this by hand. Instead, we turned to evolutionary computation to help learn the parameters for the problem. Evolution was performed with a simple ES$(2 + 10)$. The physicomimetic parameters were encoded as real values in the range $[0.0, 1.0]$ and mapped to the ranges shown in Table 1. An adaptive Gaussian mutation was used, where $\sigma \in [0.005, 0.2]$, initialized at 0.2. The ES optimized two equally weighted measures: the average incursion distance of intruders into a defense perimeter of radius 150, scaled to the range $[0.0, 1.0]$, and the hit ratio of the intruders on the resource. These measures were evaluated using five discrete-time, 350×350 continuous space simulations of the problem run for 1000 steps each. This time is sufficient to allow approximately 20 intruder attack runs per simulation. The simulation was implemented with MASON, a multiagent simulation library [11].

Table 1. Legal physicomimetic parameter ranges for resource protector agents

C_{uv}	R_{uv}	G_{uv}	d_{uv}	a_{uv}	m_u	c_{f_u}
$[0, 350]$	$[0, C]$	$[0, 2400]$	$[-5, 5]$	$[0, 5]$	$[0.1, 50.0]$	$[0, 1]$

We performed 10 independent evolutionary runs, each for 100 generations, and tested the final best parameter set for an additional 100 simulations. The resulting average scaled incursion distance measure and 95% confidence interval for this solution was 0.120 $\pm$ 0.0016, and it allowed 4 hits on the resource over the 100 simulations. As hoped, the ES took advantage of the generalized physicomimetic framework byevolving a heterogeneous solution in which 6 protectors formed an outer ring to block incoming intruders as far away from the resource as possible, while 3 protectors formed a tight cluster around the resource to block any intruders making it through the outer defense. However, the inner ring of defenders was too close to the resource to be physically plausible. The majority of the other evolutionary runs produced physically implausible solutions as well. Furthermore, all the evolved solutions had an unnatural jitter that would not be acceptable if deployed.

A more carefully considered EA might have produced more natural and physically plausible solutions in this case. Additionally, it is clear that some kind of representational constraints are necessary if one wishes to increase the level of heterogeneity: the parameter space scales quadratically as the number of particle types increases. Such considerations are attempts to implicitly add domain knowledge into the algorithm. We detail an approach that addresses the scale-up problem while allowing the engineer more control over the final solution by *explicitly* incorporating domain knowledge in the design process. Our approach is meant to *complement* the learning algorithm, though for our example problem it is sufficient to allow us to develop solutions by hand.

4 Engineering Physicomimetic Solutions Using Directed Graphs

Our goal is to systematically design formation-oriented, collaborative multiagent teams capable of true heterogeneity and modularity. While generalized physicomimetics is capable of *representing* such solutions, it isn't clear how to *design* them.

It isn't a trivial problem. The parameter space of generalized physicomimetics, in which any level of heterogeneity of team members is possible, is very large. Every agent could be represented by a different particle type. Hence, due to pair-wise interactions, the parameter space can grow quadratically with the number of types. Moreover, since there can be strong non-linear influences between these parameters, designing solutions will become increasingly intractable as the level of heterogeneity increases. Finally, with this system it is unclear how to share successful partial solutions.

We provide a principled and practical method of engineering solutions using generalized physicomimetics by noticing two key facts: We do not always need every possible interaction, and we can often reuse an interaction's parameters. Reasoning about the types of interactions is necessary for designing successful heterogeneous, swarm-based multiagent solutions. Digraphs are natural and useful tools for this type of reasoning.

4.1 A Graph-Based Force Interaction Model

Let each type of particle be a node in a digraph and each interaction be a directed edge in that graph. An edge is associated with a force law as follows: for two particle types, u and v, a directed edge $(u, v, \mathcal{F}_{uv})$ denotes an interaction where particles of type u *impart* a force on particles of type v according to the force law defined by $\mathcal{F}_{uv}$. Fig. 2 illustrates a graph for a two-agent example.

These digraphs can have isolated nodes and cycles. Omitted edges imply there is no direct interaction between the particle types represented by those nodes in the graph.

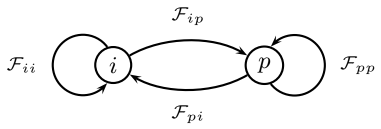
$$\begin{tikzcd} i \arrow[loop left, "\mathcal{F}_{ii}"] \arrow[r, bend left, "\mathcal{F}_{ip}"] & p \arrow[l, bend left, "\mathcal{F}_{pi}"] \arrow[loop right, "\mathcal{F}_{pp}"] \end{tikzcd}$$

Fig. 2. An example force interaction digraph. There are two particle types, (i) and (p), and there are separate force laws between every possible pair of particle types.

4.2 Modularity Via Condensed Subgraphs

In swarm engineering, the concept of modularity is particularly important. Here, we address two different views of modularity: modularity of design and behavioral modules.

When designing something complex, engineers often decompose systems, build components separately, and then combine them. We employ a similar idea for constructing complex multiagent simulations. Using our graph-based approach, we break the graph into relevant subgraphs, and then consider them in conjunction with one another. It is helpful to categorize agents by developing subgraphs that *profile* how agents of a group interact with other agents in the system. This constitutes *modular design*.

In addition to modular design, there may be times when modularizing *behaviors* (sharing subsets of behaviors) in a heterogeneous multiagent team is important. One way to introduce modularity to generalized physicomimetics is to allow particles to share force laws and parameters. We do this by allowing the engineer to *condense* a subgraph by consolidating particle types into a single node.

Some simple notational elements can be added to the digraph to aid with these sorts of design issues. This is illustrated in the next section.

4.3 A Simple Graph-Based Solution

Our generalized physicomimetic solution to the resource protection problem was versatile, but yielded a large parameter space that was quadratic with respect to the number of protector types. Careful analysis, however, reveals obvious ways that engineer-guided knowledge can limit the space in order to craft a solution to the problem by hand.

We begin our design by considering the agent types: an arbitrary number of protector types $(p_1, p_2, \ldots)$, one intruder type (i), and a resource type (r). Next we consider the types of interactions that we believe will be necessary. Since intruders cannot distinguish types of protectors, we can condense some of the intruder behaviors. Moreover, if we consider each protector type as nearly independent, we need provide only limited interactions between types of protectors — just enough to avoid hitting one another. Both of these pieces of knowledge lead to fairly obvious reductions in the model.

We designed the interactions using three subgraphs (see Figure 3), profiling protectors (all types) separately from intruders (one type). The first subgraph represents a module of behaviors for the intruder, while the second two represent two modules of behaviors for the protector types. The notation p_* in a node means all protector types are represented by that node. Links connecting such nodes represent identical force laws between the nodes. Additionally, rather than drawing many subgraphs for each type of protector, we abbreviate the design using the p_j notational convenience. Our designsolution for the interactions can be seen below. We omit the $\mathcal{F}$ labels in the graph since they are implied by the existence of the edge and identified by the nodes they connect.

Notice that we must resolve a notational conflict. The middle subgraph shows a specific edge between a particle protector type and itself, while the third subgraph shows a general edge between any two protector types. A specific edge has precedence over a general one, so the way to read the graph is as follows. Every interaction between different protector types is identical, except for the interaction of the protector type with itself — that is specified explicitly and is different for each type.

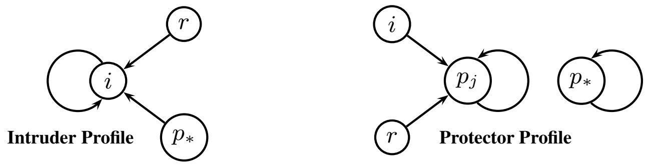

Fig. 3. Force interaction models for the resource protection problem. The graph on the left represents all the interactions affecting the intruders. Note this does not depend on the protector type. The two on the right represent those affecting the protector types. Each type of protector can react to its own type and to intruders in a different way, but they react identically to all other protectors.

We have designed our subgraphs in such a way as to capture both senses of modularity. The first and third subgraphs *condense* reactions toward any kind of protector, creating reusable modules. The second subgraph provides a design-level modularity. By using this visualization, we can compute the number of required parameters for the protector profile. Each interaction requires five parameters (G, C, R, a, and d). If there are P protector types, then $3P+1$ edges (interactions) must be defined, requiring $5(3P+1)$ parameters. Each protector type requires two more parameters (m and c_f), resulting in $2P$ additional parameters. Hence, for P protector types, $17P + 5$ parameters must be optimized. This means there will be a constant number of new parameters (17) with the addition of each protector type: a linear scaling of parameters.

With the above interaction design, it was easy to construct a solution to the resource protection problem by hand. Beginning with the first protector type, we adjusted the parameters such that these agents form a large ring around the resource. They attempt to maintain formation, but will chase off or destroy intruders that come close to them. The rest of the ring will redistribute if a protector is pulled away in pursuit of an intruder. Next we designed the second protector type to stay close to the resource, but aggressively pursue intruders that are moderately far from them. These protectors are pulled back to the resource if they get too far away, but are given a fair amount of latitude to pursue enemies that are in close quarters. In this case, combining these two behaviors was trivial — we merely sought to keep them out of the way of one another. The combined behaviors are smooth, easily understood, and physically plausible.

The parameter values for the above solution are shown below. We ran this model of 100 independent simulations; the resulting average scaled incursion distance and confidence interval was 0.199 ± 0.004. Of the 100 trials, 94 of them resulted in runs where no intruder ever struck the resource. The remaining six admitted just a single strike each.

Table 2. Model parameters for hand-coded solution to the resource protection problem

	$i \rightarrow p_1$	$r \rightarrow p_1$	$p_1 \rightarrow p_1$	$i \rightarrow p_2$	$r \rightarrow p_2$	$p_1 \rightarrow p_2$	$p_* \rightarrow p_*$
C	80	350	250	150	350	300	20
R	5	100	110	5	15	200	20
G	2400	600	1200	2400	0.05	1200	1200
d	2	2	1.5	2	-0.5	2	2
a	0	1	1	0	0	1	1

	i	p_1	p_2	r
m	1.0	1.0	1.0	60.0
c_f	0.15	0.15	0.15	-

5 Related Swarm Engineering Work

Swarm engineering, the process of designing, building, and validating swarm behaviors, has sparked much interest of late. In a recent survey of case studies applying conventional engineering approaches for dependability to swarm design, Winfield *et al.* [12] point out the need for better tools for swarm engineering.

In response to such needs, Kazadi [13] developed a formalism of *swarm intelligence* and described an approach to engineering the behavior of swarms according to that formalism. Chang [14] describes this approach as a *middle-meeting method* that combines both top-down macroscopic with bottom-up microscopic swarm design techniques. While the method provides guidance to the swarm designer in decomposing the swarm engineering problem, low-level behaviors must still be created by the designer.

In work contemporaneous with the initial development of physicomimetics, Reif and Wang [15] develop a method called *social potential fields* as a way to program large teams of robots. Like generalized physicomimetics, their method models the agents as particles, provides for multiple types of agents, and generates behaviors through interactions of forces between agents. Reif and Wang propose a hierarchical methodology for determining the set of potential force laws, laying out a step-by-step procedure for developing system behaviors with different levels of interactions.

Both Kazadi and Reif and Wang proffer methodologies for designing interactions between agents. However, both methods largely leave it to the designer to determine how to discover or create behaviors that achieve the *global goal* (Kazadi) or *required behaviors* (Reif and Wang). Our work complements these approaches to force law design by presenting an intuitive graph-based means for designing such behaviors while incorporating some human knowledge into the design process.

6 Conclusions and Future Work

This paper presented two key components of a principled method for constructing swarms in a modular way, capable of both shared and specialized behaviors using physicomimitics. We responded to the growing need to find a middle ground between open-ended, knowledge-poor representations and brittle, knowledge-rich representations by illustrating how *some* engineer-guided knowledge can be incorporated into a multiagent system. Our intent is to provide one view on how to practically develop complex swarm-based solutions in a principled way.

First, we clarified how physicomimetics can be generalized to extend the power of multiagent systems by specializing particles and their interactions. We advocate making such choices explicitly a part of the design process in constructing swarm-based systems. This gives one control over the ability of the system to produce specialized, coordinated behaviors. We illustrated these points using a challenging multiagent resource protection problem. The representational power of the generalized physicomimetic solution is more than sufficient to solve the problem; however, the scale of the parameter space necessitated heuristic optimization. This resulted in specialized squads of agents that effectively protected a central resource from intrusion, but sacrificed predictability and physical plausibility.

Second, we presented a graph-based method for designing interaction models in physicomimetic systems. This method allows engineers to construct graphs that clearly define what interactions are possible. By using our technique for condensed subgraphs, engineers can think more modularly about the design process and produce reusable behavioral modules, giving the engineer the ability to directly control the scalability of the system. We illustrated this method by hand designing a heterogeneous, modular solution to the aforementioned resource protection problem. Our solution is easy to understand, physically plausible, and performs quite well on the task.

Our next step is to apply our method to design swarm-based solutions to well-known multiagent problems, such as the art gallery problem, multi-asset surveillance, and problems from the search and rescue domain. We are also interested in combining our graph-based design method with heuristic optimization methods, designing the force interaction models by hand and eliciting the model parameters algorithmically.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence. Oxford University Press (1999)
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press (2004)
3. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers (2001)
4. Kube, C.R., Zhang, H.: Collective Robotics: From Social Insects to Robots. Adaptive Behavior **2** (1994) 189–218
5. Spears, W., Gordon, D.: Using Artificial Physics to control agents. In: IEEE International Conference on Information, Intelligence, and Systems, IEEE (1999) 281–288
6. Spears, W., Spears, D., Hamann, J., Heil, R.: Distributed, physics-based control of swarms of vehicles. Autonomous Robots **17** (2004) 137–162
7. Howard, A., Mataric, M., Sukhatme, G.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: Sixth International Symposium on Distributed Autonomous Robotics Systems. (2002)
8. Spears, W., Spears, D., Heil, R.: A formal analysis of potential energy in a multiagent systems. In: Lecture Notes in Artificial Intelligence. Volume 3228., Springer (2004)
9. Hettiarachchi, S., Spears, W., Kerr, W., Zarzhitsky, D., Green, D.: Distributed agent evolution with dynamic adaptation to local unexpected scenarios. In: Second GSFC/IEEE Workshop on Radical Agent Concepts, Springer (2006)
10. Hettiarachchi, S., Spears, W.: Moving swarm formations through obstacle fields. In: International Conference on Artificial Intelligence. Volume 1., CSREA Press (2005) 97–103
11. Luke, S., Balan, G.C., Panait, L., Cioffi-Revilla, C., Paus, S.: MASON: A java multi-agent simulation library. In: Agent 2003 Conference on Challenges in Social Simulation. (2003)
12. Winfield, A., Harper, C., Nembrini, J.: Towards dependable swarms and a new discipline of swarm engineering. In: 2004 SAB Swarm Robotics Workshop. (2004) 133–148
13. Kazadi, S.: On the Development of a Swarm Engineering Methodology. In: IEEE Conference on Systems, Man, and Cybernetics, IEEE (2005) 1423–1428
14. Chang, K., Hwang, J., Lee, E., Kazadi, S.: The Application of Swarm Engineering Technique to Robust Multi-chain Robot System. In: IEEE Conference on Systems, Man, and Cybernetics, IEEE (2005) 1429–1434
15. Reif, J.H., Wang, H.: Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots. Robotics and Autonomous Systems **27** (1999) 171–194