Experiments in the Coordination of Large Groups of Robots

Leandro Soriano Marcolino and Luiz Chaimowicz

VeRLab - Vision and Robotics Laboratory
Computer Science Department - UFMG - Brazil
{soriano, chaimo}@dcc.ufmg.br

Abstract. The use of large groups of robots, generally called *swarms*, has gained increased attention in recent years. In this paper, we present and experimentally validate an algorithm that allows a swarm of robots to navigate in an environment containing unknown obstacles. A coordination mechanism based on dynamic role assignment and local communication is used to help robots that may get stuck in regions of local minima. Experiments were performed using both a realistic simulator and a group of real robots and the obtained results showed the feasibility of the proposed approach.

1 Introduction

Cooperative robotics has become an important and active research field in the last couple of decades. Fundamentally, it consists of a group of robots working cooperatively to execute various types of tasks in order to increase the robustness and efficiency of task execution. The use of multi-robot teams brings several advantages over single robot approaches. Firstly, depending on the type of the task, multiple robots can execute it more efficiently by dividing the work among the team. More than that, groups of simpler and less expensive robots working cooperatively can be used instead of an expensive specialized robot. Robustness is also increased in certain tasks by having robots with redundant capabilities and dynamically reconfiguring the team in case of robot failures.

A natural evolution of this paradigm is the use of large groups of simpler robots, generally called swarms. Inspired by their biological counterparts, swarms of robots must perform in a decentralized fashion using limited communication. Normally these groups have to work in dynamic, partially-observable environments which increase the challenges in terms of coordination and control.

In [1] we proposed an algorithm that allows a large group of robots to overcome local minima regions while navigating to a specific goal in environments containing unknown obstacles. A coordination mechanism, based on dynamic role assignment and local communication was used to deal with robots stuck in local minima. In this paper, we experimentally validate this algorithm using both a realistic simulator and a group of real robots. We also describe and analyze the communication chain mechanism, one of the key features of the algorithm that is responsible for spreading feasible path information among team members.

2 Related Work

The general area of motion planning for large groups of robots has been very active in the last few years. One of the first works to deal with the motion control of a large number of agents was proposed for generating realistic computer animations of flocks of birds (called *boids*) [2]. In the robotics community, the more classical approaches for planning the motion of groups of robots have generally been divided into centralized and decentralized. Centralized planning consists of planning for the entire group, considering a composite configuration space. It normally leads to complete solutions but becomes impractical as the number of robots increases due to the high dimensionality of the joint configuration space. On the other hand, decentralized approaches plan for each robot individually and later try to deal with the interactions among the trajectories. This reduces the dimensionality of the problem, but can result in a loss of completeness.

A common decentralized approach for motion planning is the use of potential fields [3], in which robots are individually attracted by the goal and repelled by obstacles and other robots. In swarms, attractive forces are generally modeled through the gradient descent of specific functions [4, 5]. Unfortunately, as in regular potential field approaches, the presence of obstacles and local repulsion forces among the robots may cause convergence problems in general gradient descent approaches, mainly when robots are required to synthesize shapes. In this context, Hsieh and Kumar [6] are able to prove convergence properties and the absence of local minima for specific types of shapes and environments. Also, special types of navigation functions can be used to navigate swarms in cluttered environments [7]. But these approaches may be hard to compute in dynamic, partially-observable environments.

Another way of avoiding the dimensionality problem is to treat groups of robots as a single entity with a smaller number of degrees of freedom and then perform the motion planning for this entity. In the work presented in [8], for example, the robots can be dynamically grouped together in a hierarchical manner using a sphere tree structure. Belta et al. [9] show how groups of robots can be modeled as deformable ellipses, and presented complex decentralized controllers that allowed the control of the shape and position of the ellipses. This approach was extended in [10] with the development of a hierarchical framework for trajectory planning and control of swarms.

Certain types of tasks may require a greater level of coordination. For example, more sophisticated task allocation must be necessary for some tightly coupled tasks. When dealing with swarms, coordination mechanisms have to scale to tens or hundreds of robots. Scalable approaches for the coordination of large groups of agents (not necessarily robots) have been proposed in [11, 12] among others. It is important to mention that most of the works that deal with swarms validate their approaches only through simulation. Few papers use a real robotic infrastructure and provide experimental results, for example [7, 13, 14].

In our approach, instead of restricting the environment or developing complex controllers and coordination mechanisms, we rely on the composition of a gradient descent controller and a simple coordination mechanism to navigate swarms

in environments containing unknown obstacles. And differently from most of the papers that deal with robotic swarms, in this paper we perform real experiments to validate our approach.

3 Swarm Navigation

3.1 Controller

In this paper, the robots must move towards and spread along a goal region in an environment containing unknown obstacles. The goal region is specified by a 2D curve S given by implicit functions of the form $s(x, y) = 0$. This implicit function can be viewed as the zero isocontour of a 3D surface $f = s(x, y)$ whose value is less than zero for all points (x, y) that are inside the S boundary and is greater than zero for all points outside the S boundary. By descending the gradient of this function and applying local repulsion forces, robots are able to reach the goal and spread along the 2D curve. Details of this controller can be found in [4]. For obstacle avoidance, we augmented this controller using a regular potential field approach: if an obstacle is detected by a robot, this obstacle applies a repulsive force that is inversely proportional to the distance between them.

Thus, considering a fully actuated robot i with dynamic model given by $\dot{\mathbf{q}}_i = \mathbf{v}_i$ and $\dot{\mathbf{v}}_i = \mathbf{u}_i$, where $\mathbf{q}_i = [x_i, y_i]^T$ is the configuration of robot i, $\mathbf{u}_i$ is its control input and $\mathbf{v}_i$ is the velocity vector, the control law used by each robot is given by:

$$\mathbf{u}_i = -\alpha \nabla f^2(\mathbf{q}_i) - C\dot{\mathbf{q}}_i - \beta \sum_{k \in O_i} \frac{1}{\mathbf{d}_{ik}} - \gamma \sum_{j \in N_i} \frac{1}{\mathbf{q}_j - \mathbf{q}_i}. \quad (1)$$

Constants α, β, γ and C are positive. The first term is the inverse of the gradient used to guide the robots towards the specified shape. The second term is a damping force. The third term is the sum of repulsive forces applied by the obstacles ($\mathbf{d}_{ik}$ is the distance vector between robot i and obstacle k). Only the obstacles that are inside robot i sensing region, represented by the set O_i, are considered in the computation of forces. The fourth term computes the repulsive interaction of a robot with its neighbors, represented by the set N_i.

Unfortunately, the sum of these forces can lead to the appearance of local minima regions. Since robots are attracted by the goal and repelled by obstacles and other robots, they can be trapped in regions where the resultant force is zero or where the force profile leads to repetitive movements (for example, continuous circular movements in a specific region). Therefore, there are no formal guarantees that the robots will converge to the desired pattern. To overcome this, we rely on swarm coordination: robots may escape from local minima with the help of their teammates, as will be explained in the next subsection.

3.2 Coordination

Our coordination is based on a mode switching mechanism, generally known in robotics as dynamic role assignment [15]. A robot can switch between different

modes (or roles) during the execution of the task. Each mode determines a different behavior for the robot and will be executed while certain internal and external conditions are satisfied.

These different modes can be better modelled by a finite state machine (FSM), in which the states and edges represent respectively the modes and the possible transitions between them. In the mechanism presented in this paper, the FSM for each robot is shown in Figure 1. It is composed of five different modes: *normal*, *trapped*, *rescuer*, *attached* and *completed*.

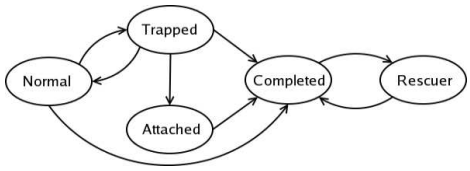


Fig. 1. Finite state machine showing the possible modes and transitions for one robot.

All robots start in the *normal* mode. A *normal* robot performs a gradient descent according to equation 1, trying to reach the goal while avoiding obstacles. It switches its mode to *trapped*, if it considers itself trapped in a local minima region. This transition is determined by the variation in the robot's position over time. If its position does not change significantly during a certain amount of time, it becomes *trapped*. A *trapped* robot may switch back to *normal* if its position start changing considerably again. It is important to note that sometimes, due to the resultant forces in the controller, robots may switch to *trapped* even if they are not in a local minima region. These false-positives do not compromise the performance of the algorithm since *trapped* robots are also controlled by equation 1, as will be explained below.

A *trapped* robot acts similarly to a *normal* one, except for the following facts: (i) a *trapped* robot strongly repels another *trapped* robot and this repulsion is stronger than the one between two *normal* robots. As a local minima region tends to attract many robots, the local interactions through these stronger repulsion forces will help some of the robots to escape this region; (ii) a *trapped* robot broadcasts messages announcing its state; (iii) *trapped* robots accept messages from *rescuers* (or *attached*) robots that will help them to escape from local minima and move towards the target. This will be better explained later in this section. We consider that all communication is local, *i.e.*, only messages received from robots within a certain distance are considered.

When a robot arrives at the target it may become a *rescuer*. Basically, when moving towards the goal, a robot saves a sequence of waypoints that is used to mark its path. If it becomes a *rescuer* it will retrace its path backwards looking for *trapped* robots. After retracing its path backwards, the robot moves again to the goal following the path in the correct direction. The number and frequency of *rescuer* robots are set empirically and may vary depending on

the total number of robots and characteristics of the environment. The method used to determine which robots become rescuers is explained in [1]. In order to minimize memory requirements, the robot discards redundant information in the path stored. Therefore, if there is a straight line in the path, ideally only two waypoints will be used.

A *trapped* robot keeps sending messages announcing its state. When a *rescuer* listens to one of these messages, thereby detecting a *trapped* robot in its neighborhood, it broadcasts its current position and its path. Any *trapped* robot will consider the message if it is within a certain distance from the *rescuer* and there is a direct line of sight between them. This restriction enables the robot to reach the rescuer's position. After receiving the message, the *trapped* robot changes its mode to *attached*.

An *attached* robot will move to the received position and then follow the received path to the goal. An *attached* robot can also communicate with other *trapped* robots, spreading the information about the feasible path to the goal. Thus, for the *trapped* robots, an *attached* robot also works as a *rescuer*. Moreover, when an *attached* robot sends a message to a *trapped* robot, it adds its current position as a new waypoint in the path information. Therefore, robots that would not have a line of sight with any *rescuer* can easily escape the local minima region thanks to the extra waypoints created in this process. As will be shown in Section 5, the *attached* robots create a powerful communication chain that allows a large number of robots to be rescued from local minima.

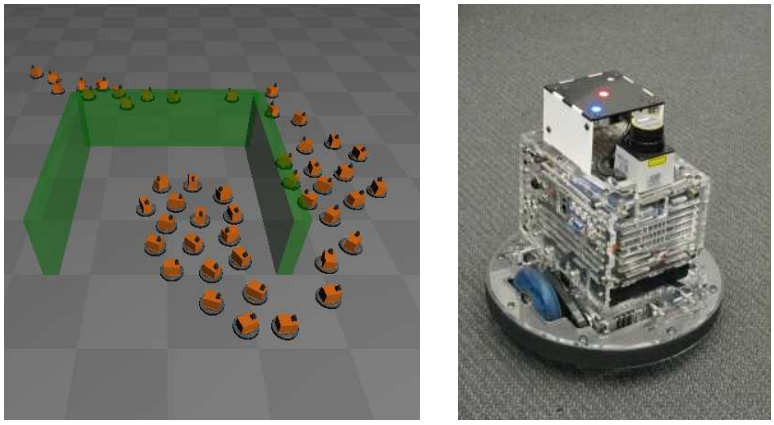
Finally, a robot will change its mode to *completed* when it reaches the target. *Completed* robots will not switch to *trapped* again but may become *rescuers* as explained above.

4 Testbed

As mentioned, one of the contributions of this paper is the validation of the proposed algorithm using both a realistic simulator and a group of real robots. In this section we present the infrastructure used for experimentation.

In terms of simulation, we used *Gazebo* [16]. Gazebo is a multi-robot simulator for both indoor and outdoor environments. It is capable of simulating a population of robots, sensors and objects in a three-dimensional world. It generates both realistic sensor feedback and physically plausible interactions between objects (it includes an accurate simulation of rigid-body physics based on ODE – Open Dynamics Engine). Figure 2(a) shows a snapshot of a simulation running on Gazebo.

Gazebo is used in conjunction with the *Player* framework [17]. Player is a network server for robot control, that provides a clean and simple interface to the robot's sensors and actuators over an IP network. It is designed to be language and platform independent, allowing the same program to run on different robotic platforms. In fact, most of the time, the algorithms and controllers used in simulation do not need to be changed to run in real robots. This specific feature is very useful for experimentation in robotics.



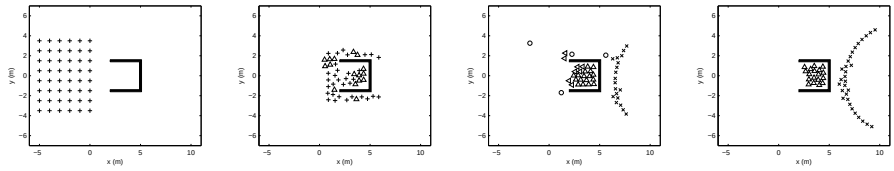
(a) (b)

Fig. 2. Testbed used in the experiments: (a) snapshot of the gazebo simulator with 48 robots and (b) one of the 7 scarab robots used in the experiments.

In the real experiments, we used a group of 7 *scarab* robots, developed in the GRASP Lab. – University of Pennsylvania (Figure 2(b)). The Scarab is a small differential drive robot equipped with an on-board computer, a Hokuyo URG laser range finder, wireless communication (802.11) and two stepper motors for actuation. The sensors, actuators, and controllers are modular and connected through the Robotics Bus (which is derived from the CAN bus protocol). A external localization system composed by a set of overhead cameras provides accurate pose information for the robots. More details about the *scarabs* and the localization system can be found in [18].

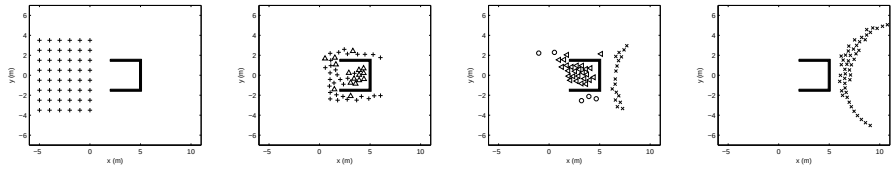
In the simulations and experiments performed in this paper, the only information provided a priori to the robots is the implicit function that attracts them to the target (function f in equation 1). Each robot knows its pose in a global reference frame, but does not have access to the pose of its teammates. This information, when needed, is explicitly transmitted by the robots using wireless communication. Also, there is no map of the environment and all obstacles are locally detected using lasers.

The laser is also used to check the existence of line of sight between a *trapped* and a *rescuer* robot. In the coordination mechanism presented in Section 3.2, *trapped* robots only accept messages from a *rescuer* if there is a line of sight between them, *i.e.*, if there is a free path connecting their positions (otherwise, it will not be able to move to the rescuer's position and then follow the feasible path to the goal). A simple algorithm, using the laser, enabled the robots to estimate the existence of line of sight. When a *trapped* robot receives a message from the *rescuer*, containing the location of the *rescuer* and a path to the target, it computes the euclidean distance (δ) and the bearing between them. Then, it turns in the direction of the *rescuer* and checks the distance returned by the laser.



(a) (b) (c) (d)

Fig. 3. Simulation of the coordination algorithm without the communication chain. Robots are represented by different shapes according to their states.



(a) (b) (c) (d)

Fig. 4. Simulation of the coordination algorithm using the communication chain.

If this distance is smaller than δ, it means that the laser detected something (an obstacle or another robot) and there is no free path between them. In this case, the *trapped* robot will ignore the message and wait for another *rescuer*.

5 Simulations

We simulated forty-eight *scarabs* navigating in an environment containing an u-shaped obstacle. This is a typical local minima scenario in robotics. In these simulations, we focus our attention on the impact of the communication chain mechanism. Other experiments, using a simpler simulator, were executed to analyze the performance of the algorithm with a varying number of robots and communication parameters in different environments. Those were previously presented in [19].

Two versions of the algorithm were tested: one without using the communication chain and the other one with this mechanism enabled. Figures 3 and 4 respectively present graphs of these two variations. In both figures, robots are represented by different shapes according to their states: *normal* ($+$), *trapped* ($\triangle$), *attached* ($\triangleleft$), *rescuer* ($\circ$), and *completed* ($\times$). Robots start on the left and the target is on the right. The u-shaped obstacle is shown in black at the middle.

In the execution without the communication chain (Figure 3), *rescuer* robots successfully reached the region where many robots were *trapped*, but only the ones in the border of the obstacle are able to escape. The other *trapped* robots did not have a direct line of sight with the rescuers and remained stuck in the local minima region. The execution with the communication chain (Figure 4), on

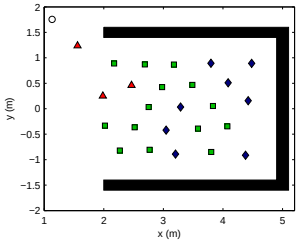


Fig. 5. Path information spread through the swarm. Shapes represent the number of message hops needed to receive the information.

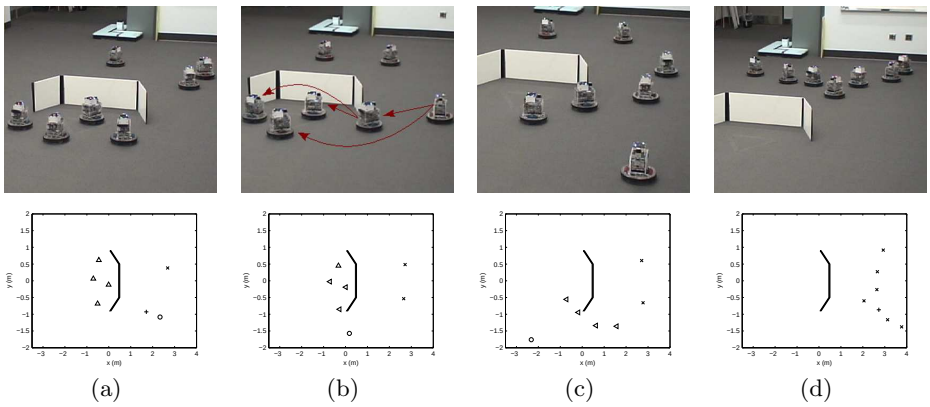
the other hand, was very effective, with all robots converging to the target. The information about the feasible path easily spread through the robots stuck in local minima, reaching even those that were closer to the bottom of the u-shaped obstacle.

In Figure 5, we take a closer look on how the information spread through the *trapped* robots when the *rescuer* (○) in the upper left sent the viable path to the goal. In this figure, shapes represent the number of message hops needed to receive the path information: triangle received with one hop, square with two hops and diamond with three hops. As can be seen, only a small number of robots, that were near the border of the obstacle, could be rescued with just one hop. Almost all robots were rescued with two hops, upon receiving a viable path from the *attached* robots. Robots that were near the bottom of the obstacle needed three hops, but could be rescued as well. As was explained, *attached* robots created new waypoints in the viable path, enabling these robots to effectively escape the local minima region.

6 Real Experiments

Real experiments are very important to show that an algorithm can effectively work in robots acting in the real world, with all the problems caused by uncertainties due to sensors and actuators errors, communication problems and real time constraints. In order to show the effectiveness of our proposed coordination mechanism, we performed real experiments with seven *scarabs* robots using a similar scenario with an u-shaped obstacle.

The sequence of snapshots of one execution can be seen in Figure 6, where the graphs on the bottom depict the robots' positions and states. The robots start in the left of the scenario and must converge to the target beyond the u-shaped obstacle in the middle. In Figure 6(a), three robots are able to move to the target, while four others are trapped in a local minima region, in front of the obstacle. The *trapped* robots are spread in this region, because of the local repulsion forces. Figure 6(b) shows a *rescuer* robot at the right of the obstacle, sending a viable path to the target. The robots that have a direct line of sight with the *rescuer* accept this message and change their status to *attached*. One of



(a) (b) (c) (d)

Fig. 6. Seven robots converge to the target with the proposed coordination mechanism. Exchange of messages is shown by the arrows.

the *attached* robots retransmits the information to the other two *trapped* robots that did not have line of sight, allowing all of them to escape the local minima region, as can be seen in Figure 6(c). Soon the state in Figure 6(d) is achieved, where almost all robots reached the target.

Thus, using the proposed algorithm, all robots effectively escaped the local minima region. Only one *rescuer* was enough to save all four *trapped* robots, thanks to the communication chain mechanism: an *attached* robot was able to spread the information to the robots that did not receive it directly.

7 Conclusion

In this paper we experimentally validated a distributed coordination mechanism for navigating a swarm of robots in environments containing unknown obstacles. Realistic simulations and real experiments with seven robots showed the viability of the proposed technique and the benefits of the communication chain.

Our future work is directed towards the improvement of the mechanism, with the development of "congestion control" techniques for the swarm. We observed that many times, when a large number of robots tried to reach the same waypoint or robots navigated in opposite directions, congestion caused by the local repulsion forces increased the time needed to reach convergence, wasting time and resources.

Acknowledgments

This work is partially supported by PRPq–UFMG, Fapemig and CNPq.

References

1. Marcolino, L.S., Chaimowicz, L.: No robot left behind: Coordination to overcome local minima in swarm navigation. In: Proc. of the 2008 IEEE Int. Conf. on Robotics and Automation. (2008) 1904–1909
2. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: Proc. of the 14th Conf. on Computer Graphics (SIGGRAPH). (1987) 25–34
3. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. Journal of Robotics Research **5**(1) (1986) 90–98
4. Chaimowicz, L., Michael, N., Kumar, V.: Controlling swarms of robots using interpolated implicit functions. In: Proc. of the 2005 IEEE Int. Conf. on Robotics and Automation. (2005) 2498–2503
5. Bachmayer, R., Leonard, N.E.: Vehicle networks for gradient descent in a sampled environment. In: Proc. of the IEEE Conf. on Decision and Control. (2002) 112–117
6. Hsieh, M.A., Kumar, V.: Pattern generation with multiple robots. In: Proc. of the 2006 Int. Conf. on Robotics and Automation. (2006) 2442–2447
7. Pimenta, L., Michael, N., Mesquita, R.C., Pereira, G.A.S., Kumar, V.: Control of swarms based on hydrodynamic models. In: Proc. of the 2008 IEEE Int. Conf. on Robotics and Automation. (2008) 1948–1953
8. Li, T.Y., Chou, H.C.: Motion planning for a crowd of robots. In: Proc. of the 2003 IEEE Int. Conf. on Robotics and Automation. (2003) 4215–4221
9. Belta, C., Kumar, V.: Abstraction and control for groups of robots. IEEE Transactions on Robotics **20**(5) (2004) 865–875
10. Kloetzer, M., Belta, C.: Hierarchical abstractions for robotic swarms. In: Proc. of the 2006 Int. Conf. on Robotics and Automation. (2006) 952–957
11. Jang, M.W., Agha, G.: Dynamic agent allocation for large-scale multi-agent applications. In: Proc. of the Workshop on Massively Multi-Agent Systems. (2004) 19–33
12. Scerri, P., Farinelli, A., Okamoto, S., Tambe, M.: Allocating tasks in extreme teams. In: Proc. of the Int. Joint Conf. on Autonomous Agents and Multiagent Systems. (2005) 727–734
13. Correll, N., Rutishauser, S., Martinoli, A.: Comparing coordination schemes for miniature robotic swarms: A case study in boundary coverage of regular structures. In: Proc. of the Int. Symposium on Experimental Robotics (ISER). (2006)
14. McLurkin, J., Smith, J.: Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In: Proc. of the 7th Int. Symposium on Distributed Autonomous Robotic Systems. (2004)
15. Stone, P., Veloso, M.: Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence **110**(2) (1999) 241–273
16. online: Gazebo Simulator. http://playerstage.sourceforge.net/gazebo/gazebo.html (acessed on March 25, 2008)
17. Vaughan, R.T., Gerkey, B.P., Howard, A.: On device abstractions for portable, resuable robot code. In: Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS 2003). (2003) 2121–2427
18. Michael, N., Fink, J., Kumar, V.: Experimental testbed for large multi-robot teams: Verification and validation. IEEE Robotics and Automation Magazine **15**(1) (March 2008) 53–61
19. Marcolino, L.S., Chaimowicz, L.: A coordination mechanism for swarm navigation: Experiments and analysis (short paper). In: Proc. of the Seventh Int. Conf. on Autonomous Agents and Multiagent Systems. (2008) 1203–1206