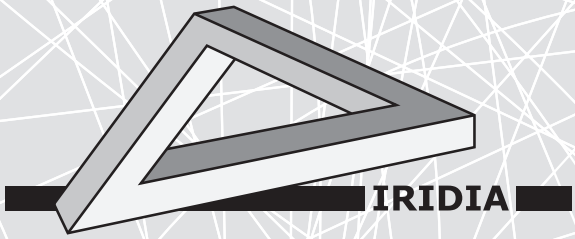


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Path Formation in a Robot Swarm

Shervin Nouyan and Marco Dorigo

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2007-002

February 2007

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2007-002

Path Formation in a Robot Swarm

February 6, 2007

Abstract

We present two swarm intelligence control mechanisms used for distributed robot path formation. In our first approach the robots form linear *chains*. We study three variants of robot chains, which vary in the degree of motion allowed to the chain structure. Our second approach is called *forcefield*. In this case, the robots form a pattern that globally indicates the direction towards a goal or home location.

We test each controller on a task that consists in forming a path between two objects which an individual robot cannot perceive simultaneously. Our simulation experiments show promising results. All the controllers are able to form paths also in complex obstacle environments, and exhibit very good scalability. Additionally, we observe that chains perform better for smaller robot group sizes, while forcefield performs better for larger groups.

1 Introduction

The capacity to navigate is a prerequisite for the accomplishment of a wide range of tasks in the robotics domain, and many different approaches have been proposed. Often, researchers equip robots with an explicit, map-like representation of their environment [1, 2]. Such a representation may be given a priori, mainly leaving the robot with the non-trivial task of self-localization, or the map may be constructed by the robot itself while moving in the environment. While this is already difficult in a static environment with a single robot, it becomes increasingly complex in dynamic environments, and in particular when multiple robots are considered. For instance, it is then necessary to distinguish between robots and obstacles, and to take into account moving objects, which considerably complicates the tasks of creating a map and self-localizing. Although solutions to such problems have been proposed [3], complex navigation strategies do not naturally scale with the number of robots, and require careful engineering of the controller in order to deal with the difficulties related to dynamic environments and multiple robots.

In this paper we are interested in approaching the navigation problem for large groups of robots following the swarm robotics principles. Swarm robotics is a growing field that emphasizes the cooperation and the collectivity of a robot

group. Rather than equipping an individual robot with a control mechanism that enables it to solve a complex task on its own, individual robots are usually controlled by simple strategies, and complex behaviours are achieved at the colony level by exploiting the interactions among the robots, as well as between the robots and the environment. When designing swarm robotics control algorithms, complex strategies are in general avoided, and instead principles such as locality of sensing and communication, homogeneity and distributedness, are followed. The main benefits that one seeks for when pursuing a swarm robotics approach are scalability with the number of robots, fault tolerance in case of individual failure, and robustness with respect to noisy conditions. These characteristics can be observed in social insects, such as ants, bees or termites, which therefore often serve as a source of inspiration.

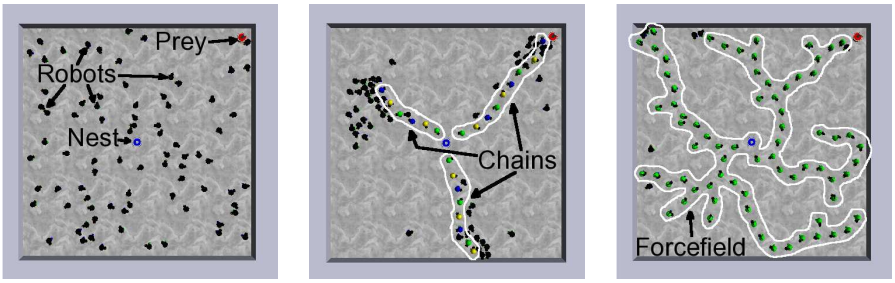
The particular navigation task we study in this paper is how to let a swarm of robots form a path between two locations in a bounded arena under the constraint that the robots' visual capacities do not allow them to perceive the two locations simultaneously. Our work is loosely inspired by the observation of ant colonies: when foraging for food, ants of many species lay trails of pheromone, a chemical substance that attracts other ants. Deneubourg *et al.* [4] showed that laying pheromone trails is a good strategy for finding the shortest path between a nest and a food source. Similarly, our robots locally manipulate the environment in order to attract other individuals and to form a global path. However, in our work the robots do not lay a substance such as pheromone. Rather, it is the robots themselves that serve as trail markers.

We propose two mechanisms, chains and forcefield, that robots employ to self-organize into visually connected structures that they use to explore and navigate the environment. Chains are linear robot structures, while in a forcefield the robots spread more evenly in space to form a tree like structure with many branches. We conducted a series of experiments in simulation to test our controllers under various conditions. We used swarms of up to 200 robots and we varied the difficulty of the task by using different distances between locations to be connected, and by using different obstacle configurations.

The remainder of this paper is organised as follows. In Section 2, we give a description of the problem under study and a brief overview of the different approaches. In Sections 3 and 4, we describe the control algorithms we used. In Section 5 we present the experimental results, and in Section 6 we discuss some related works. Finally, in Section 7 we draw some conclusions.

2 The Task and the Approach

The task that we have chosen as test-bed to analyse our control algorithms is illustrated in Figure 1. A group of robots has to form a path between two objects—denoted as nest and prey. The robots have no a priori knowledge about the dimensions and the position of any object within the environment, and a robot's perception range is small when compared to the distance between the nest and the prey.



(a) (b) (c)

Figure 1: Simulation snapshots from the initial situation (a), and a typical outcome when employing the chain (b) and the forcefield (c) controllers. 80 robots are indicated by small black circles. The task is to form a path between the blue nest in the centre of the arena, and the red prey in the top right corner. No obstacles are employed. When a robot in a chain or in a forcefield perceives the prey, a path is formed that can be used to transport the prey to the nest.

Initially, as shown in Figure 1a, all robots are placed at random positions. They search the nest, and once they perceive it, they start to self-organize into chains (Figure 1b) or into a forcefield (Figure 1c), where robots act as trail markers and attract other robots. Neighbouring robots within the path forming structure have to be able to sense each other in order to assure the connectivity. As the robots have no knowledge about the position of the prey, the structures are oriented in random directions. A self-organized process in which robots leave the structure and join it again at a different position leads to a continuous exploration of the environment until the prey is perceived. A path is then formed, and can be used by other robots to navigate between the nest and the prey, or to transport the prey to the nest. When controlled by the chain mechanism, robots in the path signal one out of three colours. The sequence of these colours gives directionality to the chain. In the forcefield controller (Figure 1c) the directionality is not given by a sequence of colours, but each robot explicitly indicates a direction. More details about the two control approaches in general, and the differences between them will be given in Section 4.

3 The S-bot and its Simulator

All the experiments presented in this paper have been conducted in simulation. Our simulation platform, called twodee, is a multi-robot simulator based on a custom high-level dynamics engine optimized for the use with the *s-bot*.[1] Figure 2a shows the physical implementation of an *s-bot*. It has a diameter of

[1]The *s-bot* was developed within the SWARM-BOTS Project, a Future and Emerging Technologies project funded by the European Commission (see `www.swarm-bots.org`).

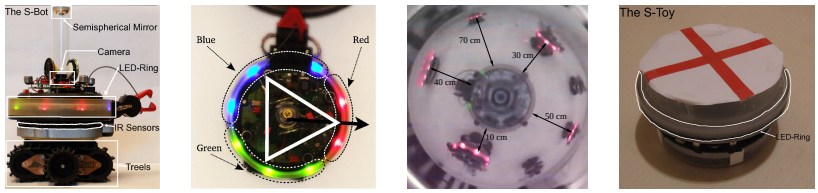
12 cm and weighs approximately 700 g. In the following, we briefly overview the actuators and sensors that we use in this study. For a more comprehensive description of the *s-bot*'s hardware see [5], and for the twodee simulator see [6].

The robot's traction system consists of a combination of tracks and two external wheels, called *treels*. For the purpose of communication, the *s-bot* has been equipped with eight RGB LEDs distributed around the robot. In particular, this LED-ring is used by robots in a chain to activate the LEDs with the colours blue, green and yellow, and by robots in a forcefield to activate a pattern which may be used to indicate a direction, as shown in Figure 2b.

The *s-bots* have 15 infra-red proximity sensors distributed around their turrets, that they use for obstacle avoidance. To allow a realistic simulation of the proximity sensors, we have recorded samples of the proximity sensor activation for various angles and distances from other objects, and we have integrated them into twodee.

A VGA camera is directed towards a spherical mirror on top of the *s-bot*, in this way providing an omni-directional view. The camera is used to perceive the nest, the prey, and other *s-bots* emitting a colour with their LED-ring. A snapshot taken from an *s-bot*'s camera is shown in Figure 2c. Due to differences among the robots' cameras, there are some variations in the perceptual ranges. The software we use to detect coloured objects allows a recognition of the red coloured prey up to a distance of $70 - 90$ cm, and of the three colours blue, green and yellow, up to $35 - 60$ cm (depending on which robot is used). Due to the spherical shape of the mirror, the distance to close objects can be approximated with good precision. For objects further away than 30 cm it becomes very difficult to deduce the distance from the camera image. The differences in the perception of the different colours and the differences between the robots are taken into account in simulation: each robot is given a different set of perceptual ranges for the four colours, and each value is chosen randomly from the ranges mentioned above.

Figure 2d shows the *s-toy*, which has a diameter of 20 cm and, as the *s-bot*,



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

Figure 2: The hardware. (a) The *s-bot*. (b) A robot activating its LEDs to indicate a direction as employed by the forcefield controller. (c) An image taken with the omni-directional camera of the *s-bot*. It shows other *s-bots* and an *s-toy* activating their red LEDs at various distances. (d) The *s-toy*, which is used as nest and as prey.

it is equipped with an RGB LED-ring. We use the *s-toy* either as nest (blue) or as prey (red). In our simulations, the nest and the prey are represented by coloured cylinders of the size of an *s-toy*.

4 Control Mechanisms

Our controllers are based on a behaviour based architecture, consisting of three states for the chain, and four states for the forcefield. Each state corresponds to a different behaviour. A behaviour is realized following the motor schema paradigm [7]. At each *control time step* only one behaviour is active.[2] For each behaviour, one or more motor schemas are active in parallel. Each motor schema outputs a vector denoting the desired direction of motion. The vectors of active motor schemas are added and translated into motor activation at each control time step.

In the following, we describe our two approaches: chains and forcefield. For each of them we first give a high-level description, and then detail the employed motor schemas, the behaviours and the conditions that trigger the transitions between the behaviours. We conclude the section by discussing the differences between the two control approaches.

4.1 Chain Controller

The robots are initially located at random positions. Typically, a robot will not perceive the nest or a chain, and therefore performs a random walk until it perceives one of them. The nest can be considered as the root of each chain. A robot that finds the nest will either start a new chain or follow an existing one. When it reaches the tail of the chain, it will join the chain with probability P_{in} per time step. Robots that are part of a chain leave it with probability P_{out} per time step, but only if they are situated at the chain's tail. The process of probabilistically joining/leaving a chain is fundamental for the exploration of the environment as it allows the formation of new chains in unexplored areas. The chain member that perceives the prey does not leave, so that when a chain encounters the prey the formed path becomes stable. At this point there are two possibilities: If the prey is closer than 30 cm, the task is successfully accomplished, and if the prey is further away than 30 cm other robots can still join the chain to make a connection that is closer to the prey.

The directionality in our chains relies on the concept of *cyclic directional patterns* (Figure 3). Each robot emits one out of three signals (i.e., LED colours) depending on its position in the chain. By taking into account the sequence of the signals, a robot can determine the direction towards the nest, or towards the prey. Figure 4 gives the state diagram of the chain controller. Each state corresponds to a robot behaviour, and arrows connecting states represent behaviour

[2] On the real *s-bot*, a control time step has a length of approximately 120 ms. We adopted the same value in simulation.

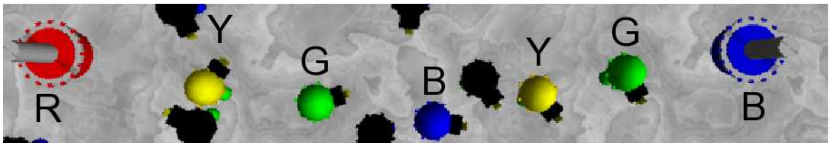


Figure 3: A chain with a *cyclic directional pattern*. The five coloured circles represent robots that have formed a chain between nest (B) and prey (R). The letters R(ed), Y(ellow), G(reen) and B(lue) denote the respective colour. Three colours are sufficient to give a directionality to the chain. A fourth colour, red, is used exclusively for the prey.

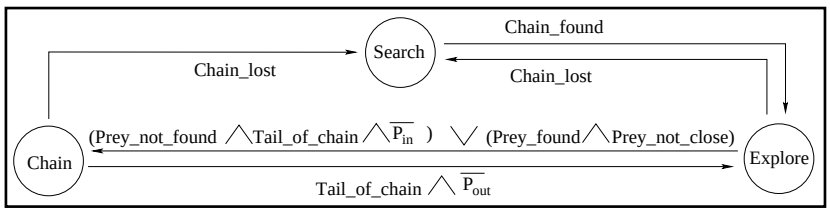


Figure 4: . State diagram of the chain controller. Each circle represents a state (i.e., a behaviour). Arrows are labelled with the conditions that trigger a state transition. The initial state is the search state. The expressions $\overline{P_{in}}$ ($\overline{P_{out}}$) are Boolean variables that are set to *true* if $R \leq P_{in}$ ($R \leq P_{out}$), and to *false* otherwise, where R is a stochastic variable sampled from the uniform distribution in $[0, 1]$, and P_{in} and P_{out} are probabilities.

transitions. The motor schemas, the behaviours and the behaviour transitions are described in the following.

Motor Schemas

- **Adjust_distance**(α, $d_{current}$, $d_{desired}$): returns a vector that points towards an object at angle α if the current distance to the object $d_{current}$ is larger than the desired distance $d_{desired}$, and in the opposite direction otherwise.

- **Move_perpendicular**(α, *direction*): returns a vector that is perpendicular to an object at angle α. The boolean parameter *direction* determines whether the vector is perpendicular in a clockwise sense or not.

- **Avoid_collisions**($IR\_sensors$): returns a vector that takes into account each activation of an IR sensor that is above a threshold.

- **Random**: returns a random vector.

- **Move_straight**: returns a vector that points forward.
- **Align**($\alpha_{previous}$, α_{next}): returns a vector that leads to the alignment between the previous and the next chain neighbour which are perceived at the angles $\alpha_{previous}$ and α_{next}.

Behaviours

- **Search**: perform a random walk. LEDs are off. Active motor schemas: Move_straight, Random, Avoid_collisions.
- **Explore**: move along a chain towards its tail or towards the nest. By default, an explorer moves towards a chain's tail. In case a robot becomes an explorer by leaving a chain, it first moves back to the nest, and then tries to follow a different chain. LEDs are off. Active motor schemas: Move_perpendicular, Adjust_distance, Avoid_collisions.
- **Chain**: the LEDs are activated with the appropriate colour, depending on the colour of the previous chain neighbour. Concerning the mobility, we employ three different strategies for chain members. (i) *Static*: no motion at all. Active motor schemas: none. (ii) *Align*: To improve the length of the chains, we implemented an alignment behaviour, that is, the robot aligns with its two closest neighbours in the chain whenever the angle between them is smaller than 120^o. Furthermore, a chain member adjusts its distance with respect to its previous neighbour to roughly 30 cm so to both avoid breaking the chain and increase the chain length. Active motor schemas: Adjust_distance, Align, Avoid_collisions. (iii) *Move*: Same as align, but if a robot is situated at the tail of a chain (i.e., if only one chain neighbour is perceived) it moves perpendicularly with respect to its neighbour, choosing randomly the direction. As the other chain members react by aligning, the net result is an overall circular movement of the chain around the nest. Active motor schemas: Move_perpendicular, Adjust_distance, Align, Avoid_collisions.

Behaviour Transitions

- **Search** → **Explore**: if a chain member is perceived. Note that the nest is perceived as a chain member, and that a robot in the search state does not react when it just perceives the prey.
- **Explore** → **Search**: if no chain member is perceived any more.
- **Explore** → **Chain**: (i) if the prey is not perceived and the tail of a chain is reached (i.e., only one chain member is perceived), the robot joins the chain with probability P_{in} per time step, or (ii) if the prey is perceived at a distance > 30 cm.
- **Chain** → **Search**: if the previous chain neighbour is no longer perceived.

- **Chain → Explore**: if a chain member is situated at the tail of a chain, it leaves the chain with probability P_{out} per time step.

4.2 Forcefield Controller

The initial situation is the same as in the previous case. The robots first have to search the nest. The first robot to find the nest stops moving and activates a colour pattern with its LEDs to point towards the nest. Another robot that perceives such a colour pattern will use the indicated direction to move away from the nest. It will end up joining to the structure of LED-activated robots when it reaches the *border*, that is, when it perceives only one LED-activated robot at a distance greater than 30 cm. The resulting robot structure can be considered as a forcefield globally leading to the nest. As can be seen in Figure 1c, the structure of the forcefield exhibits stronger branching than the chain controller (Figure 1b), because the forcefield controller employs a different rule, which allows a robot to join the structure at various positions, and not only at the tail.

The process of leaving the forcefield is probabilistic. At each time step a robot leaves the forcefield with probability P_{out}. Similarly to the chains, the process of joining/leaving the forcefield leads to a continuous exploration of the environment until the prey is found. The part of the forcefield that perceives the prey does not leave any more, so that the established path becomes stable. Again, there are two possibilities: If the prey is closer than 30 cm, the task is successfully accomplished, and if the prey is further away than 30 cm, then other robots can still join the forcefield.

When a robot leaves the forcefield it starts a random walk during which it does not react to the perception of the forcefield. Due to the random walk it might reach a different branch, stay in the vicinity of the same branch, or loose contact to the forcefield completely. The time it remains in this state is determined by the probability P_{in}. When the robot enters the search state again, it continues the random walk until it perceives the forcefield. This process is continued until the forcefield encounters the nest and in this way forms a path. Figure 5 shows a sequence of robots forming such a path. The arrows on top of the robots indicate the directions they are pointing to. The state diagram of the forcefield controller is given in Figure 6. The motor schemas, the behaviours and the behaviour transitions are detailed in the following.

Motor Schemas In addition to the motor schemas employed for the chain controller, only the following motor schema is required for the forcefield:

- **Follow_forcefield(***indicated_directions***)**: takes into account directions as indicated by all perceived robots in the forcefield, and returns a vector that points in the opposite direction.

Behaviours

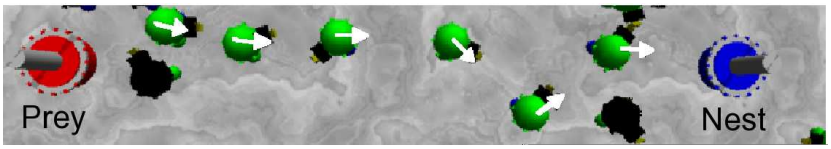


Figure 5: A forcefield. The six robots with an arrow on their top represent a path between nest and prey. The arrows indicate the directions the robots are pointing to and that lead towards the nest.

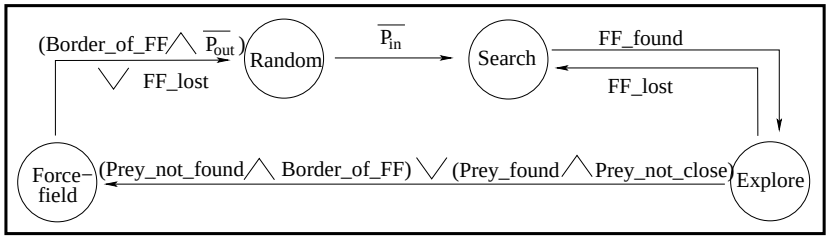


Figure 6: . State diagram of the forcefield controller. Each circle represents a state (i.e., a behaviour). Arrows are labelled with the conditions that trigger a state transition. The initial state is the search state. The expressions $\overline{P_{in}}$ ($\overline{P_{out}}$) are Boolean variables that are set to *true* if $R \leq P_{in}$ ($R \leq P_{out}$), and to *false* otherwise, where R is a stochastic variable sampled from the uniform distribution in $[0, 1]$, and P_{in} and P_{out} are probabilities.

- **Search**: perform a random walk (until the forcefield is perceived). LEDs are off. Active motor schemas: Move_straight, Random, Avoid_collisions.
- **Explore**: follow the forcefield away from the nest. LEDs are off. Active motor schemas: Follow_forcefield, Avoid_collisions.
- **Forcefield**: do not move. The LEDs are used to activate a pattern which indicates the direction towards the precedent robot in the forcefield, or towards the nest. Active motor schemas: none.
- **Random**: perform a random walk (even if the forcefield is perceived). LEDs are off. Active motor schemas: Move_straight, Avoid_collisions, Random.

Behaviour Transitions

- **Search** → **Explore**: if the forcefield is perceived. Note that the nest is perceived as part of the forcefield, and that a robot searching for the nest does not react when it just perceives the prey.

- **Explore** → **Search**: if the forcefield is no longer perceived.
- **Explore** → **Forcefield**: (i) if the prey is not perceived and only one forcefield robot is perceived at a distance $> 30\,\text{cm}$, or (ii) if the prey is perceived at a distance $> 30\,\text{cm}$.
- **Forcefield** → **Random**: (i) if the robot is situated at the border of the forcefield, it leaves the forcefield with probability P_{out} per time step, or (ii) if the forcefield is no longer perceived.
- **Random** → **Search**: with probability P_{in} per time step.

4.3 Differences between Chain and Forcefield Controllers

There are three main differences between the chains and the forcefield. First, the very nature of the signal in the structure is different. In the case of chains a direction can only be deduced when seeing at least two members of the structure, whereas in a forcefield each member explicitly broadcasts a direction. Second, the process of joining the path forming structure is probabilistic for the chains, while it is deterministic for the forcefield as robots immediately join the forcefield when they reach its border. This in general leads to a higher degree of branching of the forcefield structure. Finally, the rule employed for leaving the forcefield leads to a higher degree of randomness because the robot performs a random walk and might loose sight of the structure, having to start the search from scratch. When leaving a chain, a robot tries to stay in the vicinity of the chain while moving back to the nest to then follow another chain. Because of this lower degree of randomness, we expect the chains to perform better when there is a low density of robots.

5 Experimental Evaluation

The goal of our experimental activity was to evaluate our controllers under different experimental conditions and to compare them. In particular, we considered environments of different levels of difficulty, obtained by changing the distance between the nest and the prey in a range between one and three meters, and by considering different obstacle configurations. We also studied the scalability of our controllers by running experiments with up to 200 robots. In the following, we specify the experimental setup, we briefly describe the methodology followed to determine good parameter sets for each controller, and we present and discuss the results obtained.

5.1 Experimental Setup

We employ a bounded arena of size $5\,\text{m} \times 5\,\text{m}$. The task consists in forming a path between two locations in the environment, the nest and the prey. The nest is placed in the centre of the arena, and the prey is placed towards one

of the corners. Obstacles are cubes with a side length of 0.5 m (therefore, one obstacle occupies 1% of the arena. An instance of the task is defined by the triplet (N, D, O), where:

- N is the robot group size,
- D is the distance between nest and prey (in meters),
- O is the number of obstacles in the environment.

The initial position and orientation of the robots as well as the positions of the obstacles, are chosen randomly.

The primary performance measure is the *completion time*, that is, the time to create a path connecting prey and nest. For practical reasons, we allow a maximum completion time of 10,000 seconds. If this time is not enough to establish a path, the trial is stopped and considered as a failure.

5.2 Parameter Choice

The overall behaviour of our controllers is a function of the two parameters P_{in} and P_{out}. To set their values, we employ the racing method by Birattari *et al.* [8, 9].[3] This method is an efficient way to determine a good parameter set: it sequentially evaluates a set of candidate parameter configurations, discarding the worst performing candidates as soon as statistical evidence is gathered against them. The process is stopped when only one candidate is left, or when the candidates have been tested on all problem instances.

In our case, a candidate configuration is a set of two values for the parameters P_{in} and P_{out}. For each of these probabilities we examined ten values defined by $0.001 * 2^x$, with $x \in \{0, 1, 2, 3, \ldots, 9\}$, resulting in 100 candidates. Candidate configurations were tested on 27 experimental setups obtained considering all the possible combinations of values for N, D, and O, with $N \in \{10, 20, 40\}$, $D \in \{2, 2.5, 3\}$, and $O \in \{0, 10, 20\}$.

We decided to consider 2700 problem instances obtained from the 27 setups initialized in 100 different ways. For each controller there were between three to five candidates left for which no statistically significant difference was found based on the completion time. Among these candidates we chose those with the highest success rates.

5.3 Results

Table 1 gives a first overview of the overall performance of the different algorithms. Under the above mentioned experimental conditions, the aligning and moving chains clearly outperform the others, and successfully form a path in most problem instances where this is possible. A success rate of 100% is not reachable because the problem mix includes tasks that cannot be solved by 10

[3] In a previous work [10] we have studied the impact of the parameters on the overall behaviour of the robot chains and compared the performance of different parameter sets.

Table 1: The selected parameter sets based on the outcome of a racing algorithm on 27 experimental setups obtained considering all the possible combinations of values for N, D, and O, with $N \in \{10, 20, 40\}$, $D \in \{2, 2.5, 3\}$, and $O \in \{0, 10, 20\}$. Each setup was initialized in 100 different ways.

Controller	P_{in}	P_{out}	Success Rate	Median Completion Time
Static Chain	0.064	0.008	62.2 %	4142 seconds
Aligning Chain	0.128	0.004	89.7 %	1066 seconds
Moving Chain	0.128	0.004	91,9 %	1181 seconds
Forcefield	0.064	0.016	48.1 %	$>$ 10000 seconds

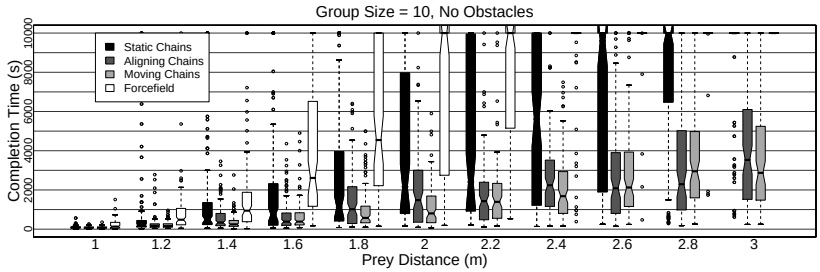


Figure 7: Completion time for changing nest to prey distance. Group of 10 robots in an environment without obstacles.

robots (in principle, 9 robots are sufficient for a prey distance of 3 m, but only if they are able to form a straight line, which might not be feasible any more when obstacles are added to the environment).

The forcefield performs worse than the others, not forming a path in 51.9% of the cases. The lower success rate of the forcefield and of the static chains is due to the following reasons. First, the structures they form do not move at all. Therefore, in general, they cover shorter distances from the nest than the two dynamic chain strategies whose structures can stretch over longer distances thanks to the aligning mechanism. Second, but only for the forcefield, the process of leaving the path forming structure induces more randomness than in the case of the chains: the robots spend more time searching the nest or a forcefield. This is a big disadvantage for small robot groups. In fact, in this case the lower robot density makes it more difficult for a robot to encounter the forcefield or a chain when performing a random walk. To assess the performance with small groups we conducted an experiment employing 10 robots in an obstacle free environment. In the experiment we measure the completion time when we change the nest to prey distance in the interval 1 to 3 meters. The results are shown in Figure 7.

The aligning and the moving chains reach high success rates for all tested distances. Confirming our above hypothesis, and as also hypothesized at the end of Section 4, the results show that indeed for a small group size the forcefield

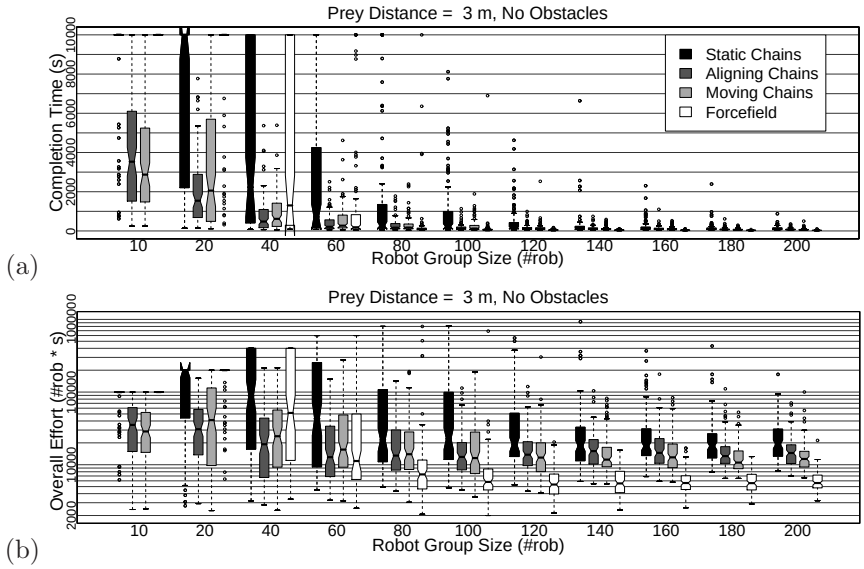
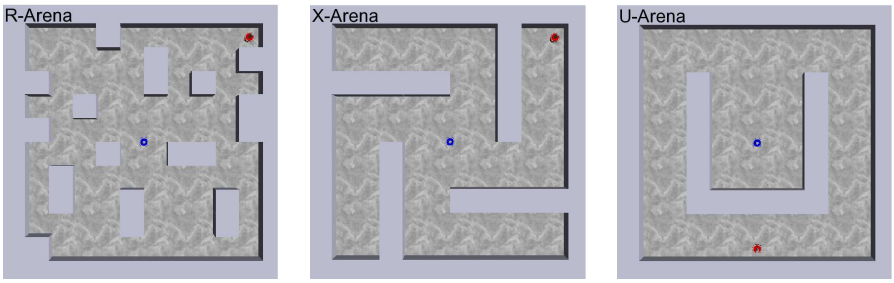


Figure 8: Scalability test in an environment without obstacles. 100 trials were conducted for each setup. (a) Completion time. (b) Overall effort (i.e., the product of completion time and group size). It measures the scalability of the system and is plotted in logarithmic scale.

performs worse than the chain strategies.

To further analyse this issue, we carried out a scalability test. In this test we kept the prey at a distance of 3 m, and again used an obstacle free environment. A summary of the results is given in Figure 8. First, in Figure 8a the completion time is shown. The performance of all controllers, and in particular of the forcefield controller, increases with the group size. While the forcefield performs rather poorly for group sizes of up to 40, from 60 robots on it reaches the same performance as the aligning and the moving chains, and for higher group sizes outperforms them. The higher amount of randomness that limits the performance for smaller groups, apparently turns into an advantage for larger groups. In fact, it allows the robots to move more freely after they have left the forcefield, in this way allowing for a more homogeneous dispersion of the robots in the environment. For increasing group sizes chains tend to become overcrowded with robots moving along them. This increases the amount of physical interactions and makes it difficult for the robots to move efficiently.

Second, Figure 8b displays the overall effort, that is, the product of completion time and robot group size. This measure is a good indicator of scalability. A decrease for growing group sizes means that the added resources lead to a more than proportional decrease in completion time. The results show that our controllers have good scalability: for all controllers the overall effort decreases



(a) (b) (c)

Figure 9: The three different types of arena used. (a) The R-arena has a random positioning of the obstacles. In this case 20 obstacles were included. (b) The X-arena has four corridors. The prey is hidden behind one of them. (c) The U-arena with a large open central place. The prey is positioned behind a U-shaped obstacle.

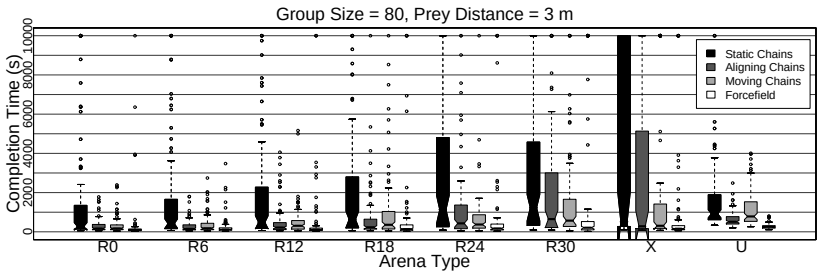


Figure 10: Obstacle test for a group of 80 robots and a prey distance of 3 meters. For arenas of type R the number of obstacles is indicated. 100 trials were conducted for each setup.

up to 100 robots, and then remains roughly unchanged.

To assess the performance in presence of obstacles, we tested our controllers on three types of obstacle environments. In addition to the standard arena with a random configuration of obstacle cubes (R-arena, Figure 9a), we also use two predefined arenas with a fixed configuration of obstacles: the X-arena (Figure 9b) and the U-arena (Figure 9c).

Figure 10 shows the results of the obstacle experiment for a group of 80 robots. The prey distance is 3 m for all cases except for the U-arena, where it is placed behind a long corridor at a distance of 2.12 m. By adding obstacles to the environment, the task becomes more difficult at several levels. First, the presence of obstacles increases the difficulty of navigation. Second, finding the nest or the prey becomes more difficult because they might be hidden behind the obstacles. Third, it might be impossible to form a straight path connecting

nest and prey. This obviously increases the length of the shortest path, and it implies a particular difficulty for the aligning and the moving strategies, which both attempt to align the chains when a certain threshold angle of the chain neighbours is surpassed.

The results show that in general our controllers are capable of coping with obstacles. Even if the completion time increases with more complex configurations, the task is solved in most cases. The static chains perform worse than the other strategies. For the considered group size, the forcefield is the most successful strategy. The presence of obstacles decreases its performance to a lesser degree than for the chain controllers. This also holds for smaller group sizes, for which we in general found similar results. However, as already shown for environments without obstacles, the forcefield performs considerably worse for smaller robot groups.

6 Related Works

As mentioned in the introduction, traditional approaches to environment navigation are often based on an internal map-like representation of the environment [1, 2]. Such approaches do not scale well for large groups of agents, where a distributed control strategy may be better suited. When approaching the problem of controlling swarms of robots, researchers often take inspiration from social insects and sometimes directly refer to the term pheromone [11, 12, 13], or to ants [14].

All these approaches employ distributed control mechanisms, and mostly use simple strategies and local information. We can roughly distinguish between two categories of distributed multi-agent path planning:

- **The path is formed by a network of immobile devices.** The devices are placed either a priori at fixed positions, or by the robots themselves. An individual network node is usually very limited in its sensing and computing capabilities. Robots can locally communicate with the network to find a path in the environment. Due to their simplicity, network nodes have low power consumption and are relatively cheap to produce, which makes them ideally suited for large scale experiments. For instance, O'Hara and Balch [15] use a sensor network with up to 156 Gnats sensor nodes that compute the shortest path using the distributed Bellman-Ford algorithm [16], and test the impact of different configurations of sensors placement. Li *et al.* [17] use a similar approach with 50 sensors of the Mote platform and take into account so called danger zones which have to be avoided. Batalin *et al.* [18] study a sensor network in the context of terrain coverage and navigation. A robot action is computed based on transition probabilities between the nodes. They use the Pioneer mobile robot and 9 nodes.

  In the simplest case, network nodes do not have any sensory capabilities at all and are used as landmarks or as a medium for indirect, so called

stigmergic, communication. A promising example for this are RFID-based devices. Mamei and Zambonelli [11] use such passively powered RFID tags in an office environment to mark fixed locations such as a door or a table, and to identify objects that may move around, such as keys or pencils. In their experiments, robots can manipulate the RFID tags and leave a trail which enables other robots to find particular objects. They encountered some problems due to the very limited storage capacity. Nevertheless, the general idea of using RFID technology is very appealing as RFID can be produced very cheaply, and will probably soon be found everywhere.

In addition to their low production cost, such devices in general have the advantage of being more robust than robots. However, they have to be placed in the environment a priori, or by the robots. This is not required if the robots form the path themselves.

- **The robots serve as landmarks or beacons themselves**. This is the case for our approach. When designing our controllers, we took inspiration from Goss and Deneubourg [19], who have studied robot chains for a prey retrieval task. In their approach, every robot in a chain emits a signal indicating its position in the chain. A similar system was implemented by Drogoul and Ferber [20]. Both works were carried out in simulation, and differ from our approach to chains because robots in a chain structure need to transmit as many signals as there are robots. This leads to an increasing degree of complexity for growing group sizes, and therefore to worse scalability characteristics. In our approach the number of different signals is independent of the number of robots. For the chains, three colours for nest and chain, and one colour for the prey are required. For the forcefield two colours for nest and prey, and one pattern for direction indication suffice.

  Werger and Matarić [21] use real robots to form a chain in a prey retrieval task. In their case the chains are not visually connected. Rather, they rely on physical contact: one robot in the chain has to regularly touch the next one in order to maintain the chain.

  Payton *et al.* [12, 13] study robot networks which can be used to represent a path as well. To build up the robot network different strategies are proposed. A gas expansion model leads to a uniform distribution similar to our forcefield. A group of robot first spreads in the environment using simple attraction/repulsion mechanisms. Afterwards the robots communicate three different sorts of pheromone to select the shortest of the many different possible paths. In our case, the network is built up incrementally, and the robots do not need to communicate at all with each other, except for indicating a direction. Another strategy to form the network is referred to as guided growth, and results in less branched structures such as our chains. One robot is selected to be the leader. The other robots follow this leader and in this way the robot structure stretches to form a line. The leading robot can for instance be designated by the user.

7 Discussion and Conclusions

We have presented an experimental study of two control approaches that employ visually connected robot structures to form a path between two objects. One of these approaches relies on the formation of linear structures, called chains. We distinguish three variants that differ by the degree of motion allowed to the chains: (i) static chains with no motion at all, (ii) aligning chains, which attempt to form a straight line, but stop moving once aligned, and (iii) moving chains, where the chains continue to move also after the alignment. The other approach, called forcefield, leads to the formation of a tree like structure with many branches. Both approaches are completely distributed and homogeneous, and make use of local information and communication only.

In an experimental study, we first determined good parameter sets, and then extensively tested each controller under various conditions. In general, our controllers reached a good performance, also in the presence of obstacles. While all controllers exhibit very good scalability characteristics, the chains perform better for groups of up to 40 robots, while the forcefield performs better for larger group sizes, starting from approximately 60 robots. Among the chain variants, the static chains are clearly outperformed by the other two strategies.

We believe that the higher degree of scalability of the forcefield controller is due to its use of simpler rules and more randomness. The chain controllers are a slightly more engineered solution, where we tried to reach a high degree of efficiency. For instance, when robots leave a chain, they move back to the nest to then follow another chain and join it. On the other hand, when leaving a forcefield, the robot starts a random walk with the risk of losing contact with the forcefield. While this may be a disadvantage for small group sizes, it turns out to work quite well for larger robot groups.

In the future we want to implement our control mechanisms on the real robots. For the chains this has already been done. In our experiments we used up to eight real *s-bot* robots, and showed how a formed path can be exploited by other robots to transport a heavy object, which can not be moved by a single robot [22]. The forcefield controller has not been implemented on the real robots yet. However, within a cooperative transport task we have already tested the mechanism of direction indication, which is fundamental to the forcefield [23].

Acknowledgments

This work was supported by the "ANTS" project, an "Action de Recherche Concertée" funded by the Scientific Research Directorate of the French Community of Belgium, and by the "SWARM-BOTS Project", funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-2000-31010. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. Marco Dorigo acknowledges support from the Belgian FNRS, of

which he is a Research Director.

References

[1] Filliat, D., Meyer, J.A.: Map-based navigation in mobile robots - I. A review of localization strategies. Cognitive Systems Research **4** (2003) 243–282

[2] Meyer, J.A., Filliat, D.: Map-based navigation in mobile robots - II. A review of map-learning and path-planning strategies. Cognitive Systems Research **4** (2003) 283–317

[3] Howard, A.: Multi-robot mapping using manifold representations. In: Proc. of the 2004 IEEE Int. Conf. on Robotics and Automation, IEEE Computer Society Press, Los Alamitos, CA (2004) 4198–4203

[4] Deneubourg, J.L., Aron, S., Goss, S., Pasteels, J.M.: The self-organizing exploratory pattern of the argentine ant. J. Insect Behavior **3** (1990) 159–168

[5] Mondada, F., Gambardella, L.M., Floreano, D., Nolfi, S., Deneubourg, J.L., Dorigo, M.: The cooperation of swarm-bots: Physical interactions in collective robotics. IEEE Robotics & Automation Magazine **12**(2) (2005) 21–28

[6] Christensen, A.L.: Efficient neuro-evolution of hole-avoidance and phototaxis for a swarm-bot. Technical Report TR/IRIDIA/2005-14, Université Libre de Bruxelles, Belgium (2005) DEA Thesis.

[7] Arkin, R.: Behavior-Based Robotics. MIT Press, Cambridge, MA (1998)

[8] Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In Langdon, W.B., et al., eds.: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann Publishers, San Francisco, CA, USA (2002) 11–18

[9] Birattari, M.: The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective. Volume 292 of DISKI. Infix, AKA/IOS Press, Berlin, Germany (2005)

[10] Nouyan, S., Dorigo, M.: Chain based path formation in swarms of robots. In Dorigo, M., et al., eds.: Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006. Volume 4150 of LNCS. Springer Verlag, Berlin, Germany (2006) 120–131

[11] Mamei, M., Zambonelli, F.: Physical deployment of digital pheromones through RFID technology. In: Proc. of the 4th Int. Conf. on Autonomous Agents and Multi-Agent Systems, AAMAS 2005. (2005) 1353–1360

[12] Payton, D., Daily, M., Estowski, R., Howard, M., Lee, C.: Pheromone robotics. Autonomous Robots **11** (2001) 319–324

[13] Payton, D.: Pheromone robotics and the logic of virtual pheromones. In Şahin, E., Spears, W.M., eds.: From Animals to Animats 8. Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB04). Volume 3342 of LNCS., Springer Verlag, Berlin, Germany (2004) 45–57

[14] Svennebring, J., Koenig, S.: Building terrain covering ant robots: a feasibility study. Autonomous Robots **116**(3) (2004) 193–221

[15] O'Hara, K., Balch, T.: Pervasive sensor-less networks for cooperative multi-robot tasks. In: Proceedings of the Seventh International Symposium on Distributed Autonomous Robotic Systems, Springer Verlag, Berlin, Germany (2004) In Press.

[16] Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, NJ (1957)

[17] Li, Q., De Rosa, M., Rus, D.: Distributed algorithms for guiding navigation across a sensor network. In: 9th International Conference on Mobile Computing and Networking, New York, NY, ACM Press (2003) 313–325

[18] Batalin, M., Sukhatme, G.S.: Spreading out: A local approach to multi-robot coverage. In: Proceedings of the Sixth International Symposium on Distributed Autonomous Robotic Systems, Fukuoka, Japan, Springer Verlag, Berlin, Germany (2002) 373–382

[19] Goss, S., Deneubourg, J.L.: Harvesting by a group of robots. In: Proc. of the 1st European Conf. on Artificial Life, MIT Press, Cambridge, MA (1992) 195–204

[20] Drogoul, A., Ferber, J.: From Tom Thumb to the dockers: Some experiments with foraging robots. In: From Animals to Animats 2. Proc. of the 2nd Int. Conf. on Simulation of Adaptive Behavior (SAB92), MIT Press, Cambridge, MA (1992) 451–459

[21] Werger, B., Matarić, M.: Robotic food chains: Externalization of state and program for minimal-agent foraging. In: From Animals to Animats 4, Proc. of the 4th Int. Conf. on Simulation of Adaptive Behavior (SAB96), MIT Press, Cambridge, MA (1996) 625–634

[22] Nouyan, S., Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Group transport along a robot chain in a self-organised robot colony. In: Proc. of the 9^{th} Int. Conf. on Intelligent Autonomous Systems, IOS Press, Amsterdam, The Netherlands (2006) 433–442

[23] Campo, A., Nouyan, S., Birattari, M., Groß, R., Dorigo, M.: Negotiation of goal direction for cooperative transport. In Dorigo, M., et al., eds.: Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006. Volume 4150 of LNCS. Springer Verlag, Berlin, Germany (2006) 191–202