Swarm Intell (2008) 2: 1–23
DOI 10.1007/s11721-007-0009-6

Path formation in a robot swarm

Self-organized strategies to find your way home

Shervin Nouyan · Alexandre Campo · Marco Dorigo

Received: 31 January 2007 / Accepted: 9 November 2007 / Published online: 6 December 2007

Abstract We present two swarm intelligence control mechanisms used for distributed robot path formation. In the first, the robots form linear *chains*. We study three variants of robot chains, which vary in the degree of motion allowed to the chain structure. The second mechanism is called *vectorfield*. 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 in complex obstacle environments and exhibit very good scalability, robustness, and fault tolerance characteristics. Additionally, we observe that chains perform better for small robot group sizes, while vectorfield performs better for large groups.

Keywords Swarm intelligence · Swarm robotics · Distributed path formation

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 (Filliat and Meyer 2003; Meyer and Filliat 2003). Such a representation may be given a priori, mainly leaving a 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. In this case, in fact, it becomes necessary to distinguish between robots and obstacles, and to take into account moving objects, which considerably complicates the tasks of creating a map and of self-localizing. Although

S. Nouyan (✉) · A. Campo · M. Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
e-mail: snouyan@ulb.ac.be

solutions to such problems have been proposed (Howard 2004), 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 tackling the navigation problem for large groups of robots following 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 obtained 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 when pursuing a swarm robotics approach are scalability with the number of robots, robustness with respect to noisy conditions, and fault tolerance in case of individual failure. 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. (1990) 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 vectorfield, that robots employ to self-organize into visually connected structures to explore and navigate the environment. Chains are linear robot structures, while in a vectorfield the robots spread more evenly in space to form a tree like structure with many branches. We conduct a series of experiments in simulation to test our controllers under various conditions. We vary the difficulty of the task by using different distances between locations to be connected and by using different obstacle configurations. Furthermore, we test our controllers for scalability with swarms of up to 200 robots, for robustness to noisy conditions by manipulating the noise of the various sensors, and for fault tolerance with respect to individual failure by completely disabling some of the sensors or actuators of some of the robots in the group.

There are two main contributions of this paper. First, we propose two novel control mechanisms. Similarities and differences from other approaches are discussed along with the related work in Sect. 6. Second, we conduct an extensive analysis to compare the two mechanisms and to show that they indeed achieve a high degree of scalability, robustness, and fault tolerance.

The remainder of this paper is organised as follows. In Sect. 2 we give a description of the problem under study and a brief overview of the two mechanisms proposed. In Sect. 3 we describe the simulation environment in which we ran our experiments, and in Sect. 4 we describe the control algorithms we used. In Sect. 5 we present the experimental results, and in Sect. 6 we discuss related work. Finally, in Sect. 7 we draw conclusions.

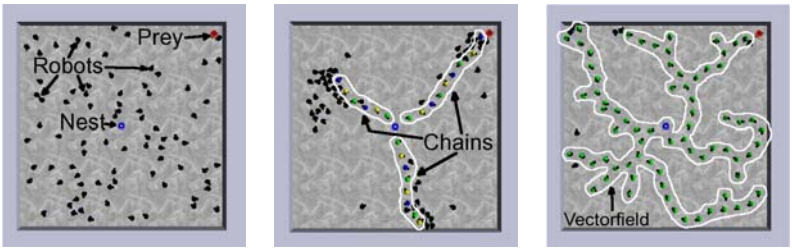


Fig. 1 Simulation snapshots from the initial situation (**a**), and a typical outcome when employing the chain (**b**) and the vectorfield (**c**) controllers. 80 robots are indicated by small *black circles*. The task is to form a path between the blue nest in the *centre* of the arena, and the red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

2 The task and the approach

The task that we have chosen as a test-bed to analyse our control algorithms is illustrated in Fig. 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. The difficulty of the task can be varied by changing the distance between nest and prey, and by placing obstacles in the environment.

Initially, as shown in Fig. 1a, all robots are placed at random positions. They search the nest, and once they perceive it, they start to self-organize into chains (Fig. 1b) or into a vectorfield (Fig. 1c). In both cases 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 discovered. 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 vectorfield controller (Fig. 1c) the directionality is not given by a sequence of colours, but each robot explicitly indicates a direction. More details about the two control mechanisms in general, as well as the differences between them, will be given in Sect. 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. It has been optimized for the use with the *s-bot*'s,[1] and controllers developed in simulation have been successfully ported to the real robot for several tasks (Christensen

[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).

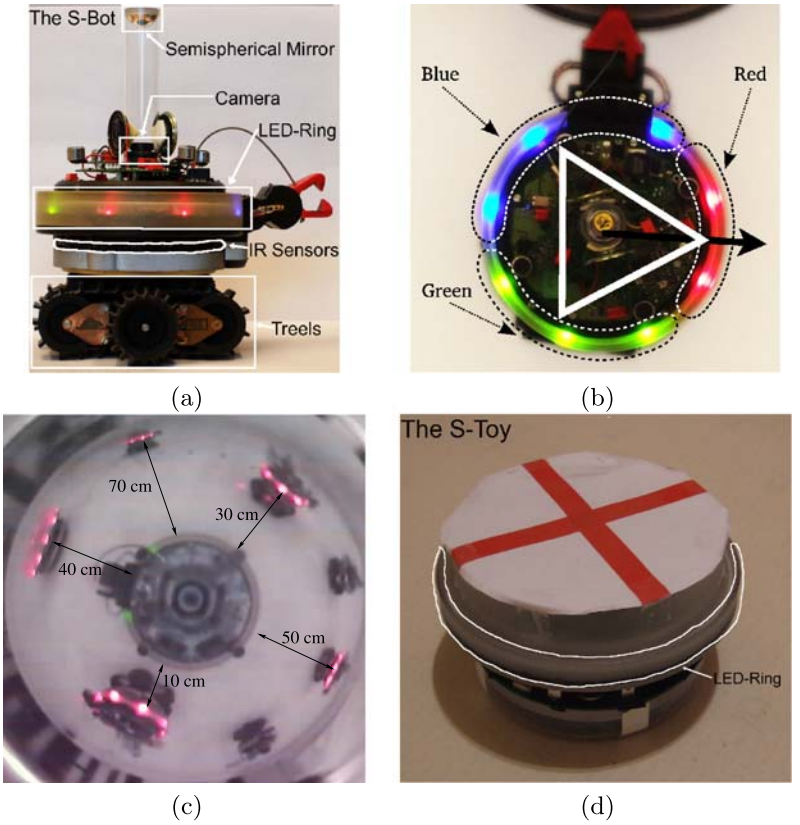


Fig. 2 The hardware. **a** The *s-bot*. **b** A robot activating its LEDs to indicate a direction as employed by the vectorfield controller. **c** An image taken with the omni-directional camera of the *s-bot*. It shows other *s-bots* and an *s-toy* activating their red LEDs at various distances. **d**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

and Dorigo 2006; Christensen et al. 2007, 2008; Nouyan et al. 2006). In this section, we give a description of the *s-bot* robot and explain the mechanisms employed to ensure a realistic simulation. For a more comprehensive description of the *s-bot*'s hardware see Mondada et al. (2005) and for the TwoDee simulator see Christensen (2005).

The s-bot robot and the s-toy Figure 2a shows the physical implementation of an *s-bot*. It has a diameter of 12 cm and weighs approximately 700 g. In the following, we briefly overview the actuators and sensors relevant to this study.

The robot's traction system consists of a combination of tracks and two external wheels, called *treels*. The *s-bot* is capable of a maximum speed of 30 cm/s. For the chaining mechanism, we used a maximum speed of 13 cm/s on the real robot. This speed corresponds to a maximum angular velocity of 97.6 deg/s when turning on the spot.

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 vectorfield to activate a pattern which may be used to indicate a direction, as shown in Fig. 2b.

In order to perceive the LED-ring, a VGA camera is mounted on top of the *s-bot* and is directed towards a spherical mirror, 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 Fig. 2c. Given that the spherical mirror is mounted at 10 cm on top of a robot, another robot does not entirely block the view of the camera. However, obstacles, such as those employed in our experiments, do block the view. Due to differences among the robots' cameras, there are some variations in the perceptual ranges. The software we use on the real robot to detect coloured objects allows 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 up to a distance of 30 cm, but it becomes increasingly difficult to deduce the distance for objects that are further away. The direction to other objects is perceived quite precisely, and the precision increases with growing distance.

The *s-bot* has 15 infra-red proximity sensors distributed around its turret and used for obstacle avoidance. Using these sensors the *s-bot* can recognize another object when its distance is less than 15 cm.

Figure 2d shows the *s-toy*, which has a diameter of 20 cm and, as the *s-bot*, is equipped with an RGB LED-ring. We use the *s-toy* either as nest (blue) or as prey (red).

The TwoDee simulator The *s-bot* is modelled as a congregate object of a cylinder with the diameter of the *s-bot* body, and of a cuboid with the dimensions of the *s-bot* gripper. The nest and the prey are represented by coloured cylinders of the size of an *s-toy*. Despite the efforts to devise a precise simulation, some characteristics of the robots and of the robot-environment interaction may escape the modelling phase. For this reason, noise is used to ensure that the behaviour developed in simulation will cope with differences between simulation and reality (Jakobi 1997; Jakobi et al. 1995). Noise is simulated for both actuators and sensors, adding a random value uniformly distributed in a given range. The noise distribution from the real robots is modelled by a uniform noise distribution. The bounds of the added random values are specified below and are in general higher than the standard error observed on the real robots.

The treels have been simulated by two active wheels. The speed of each wheel is set individually. We adopted the values of maximum speed and angular velocity as reported above. When setting the speed of a wheel to v_0, we add a noise value in the range $[-0.1 \cdot v_0; 0.1 \cdot v_0]$.

The camera and the proximity sensors have been modelled in the simulator trying to closely match their physical counterpart. A sampling technique was employed using samples from the corresponding devices recorded from the real robot (Miglino et al. 1995). These samples are collected in a matrix of activation values that can afterwards be used in the simulation to characterise the sensor activation for a given situation.

For the camera, we recorded 100 samples from camera images for 36 angles and 22 distances in the range [5 cm; 100 cm] with respect to either a prey, a nest, or another *s-bot*. When calculating distance and direction to another object in simulation, we take the median values from the collected samples for the given situation, and add noise values in the ranges [–10 cm, 10 cm] for the distance and $[-18°, 18°]$ for the direction. Concerning the perception of LED patterns indicating a direction, as used by the vectorfield, we add a noise value in the range $[-36°, 36°]$ to the median value taken from the samples. The differences in the perception of the different colours and the differences between the robots are taken into account in simulation as well: 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.

The proximity sensors, like the camera, have been modelled by recording 100 samples from the proximity sensor activation for 36 angles and 18 distances in the range [1 cm;

20 cm] with respect to either another *s-bot* or to a wall. To calculate the value of a proximity sensor in simulation, we take the median value from the collected samples for the given situation, and add a noise value in the range $[-0.2 \cdot prox_{\max}, 0.2 \cdot prox_{\max}]$, where $prox_{\max}$ is the proximity sensor saturation value.

4 Control mechanisms

Our controllers are based on a behaviour-based architecture, consisting of three states for the chain, and four states for the vectorfield. Each state corresponds to a different behaviour. A behaviour is realized following the motor schema paradigm (Arkin 1998). At each 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 weighted sum of the vectors of the active motor schemas is translated into motor activation. The weights are set by trial and error.

In the following, we describe our two mechanisms: chains and vectorfield. 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 their differences.

4.1 Chain controller

The robots are initially located at random positions. Typically, at the beginning a robot does not perceive the nest or a chain, and therefore performs a random walk until it perceives one of them. The robot will not react if it perceives the prey because no path has been formed yet. 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 a 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 sets its value of P_{out} to zero, 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, while 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.

We implemented three variants of chains, which vary in the degree of motion allowed to the chain structure. In the simplest case, referred to as static, there is no motion at all. In the second case, referred to as align, the chains as a whole align. Finally, in the third case, the chains perform a circular movement around the nest.

The directionality in our chains relies on the concept of *cyclic directional patterns* (Fig. 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 transitions. The motor schemas, the behaviours and the behaviour transitions are described in the following.

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

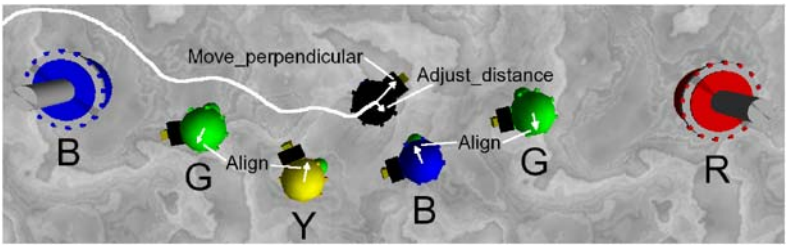


Fig. 3 A chain with a *cyclic directional pattern*. The four coloured *circles* represent robots that have formed a chain between nest (B) and prey (R). The letters R(ed), Y(ellow), G(reen), and B(lue) denote the respective colour. Three colours are sufficient to give a directionality to the chain. A fourth colour, red, is used exclusively for the prey. The *four vectors* drawn on top of the chain members represent the motor schema that leads to an overall alignment of the chain. The *two vectors* drawn on top of the explorer robot represent the motor schemas that lead to a tangential trajectory, as shown by a *white line* denoting the path followed by the robot

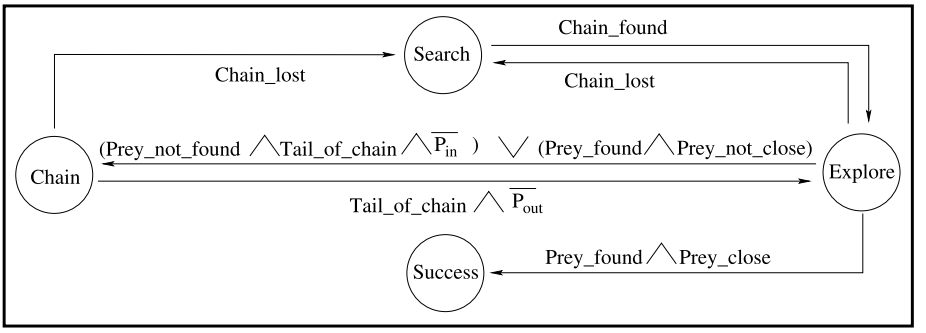


Fig. 4 State diagram of the chain controller. Each *circle* represents a state (i.e., a behaviour). *Arrows* are labelled with the conditions that trigger a state transition. The initial state is the search state. The expressions $\overline{P_{\text{in}}}$ ($\overline{P_{\text{out}}}$) are Boolean variables that are set to *true* if $R \leq P_{\text{in}}$ ($R \leq P_{\text{out}}$), 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

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. The length of the returned vector is proportional to the value of $\|d_{\text{current}} - d_{\text{desired}}\|$. In order to avoid an oscillating behaviour, the vector is set to zero if $\|d_{\text{current}} - d_{\text{desired}}\| < 5$ cm.
- *Move_perpendicular*(α, *clockwise*): returns a unit vector that is perpendicular to an object at angle α. The boolean parameter *clockwise* 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. The direction of the vector is opposite to the direction of the sensor with maximum activation, and its length is proportional to the difference between the activation and the threshold.
- *Random*: returns a random unit vector.
- *Move_straight*: returns a unit vector that points forward.
- *Align*(α_{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 α_{previous} and α_{next}. The

length of the vector is proportional to the value of $180° - |\alpha_{\text{previous}} - \alpha_{\text{next}}|$. In order to avoid an oscillating behaviour, the vector is set to zero if $|\alpha_{\text{previous}} - \alpha_{\text{next}}| > 170°$ (with 180° representing perfect alignment).

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 turns around the nest to follow a (possibly) different chain or probabilistically decides to start a new chain by itself. LEDs are off. An example of the trajectory of an explorer moving along a chain is given in Fig. 3. 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 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°. 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 a random 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.
- *Success*: the experiment is successfully finished. No action is performed.

Behaviour transitions

- *Search* → *Explore*: if a chain member is perceived. Note that the nest is perceived as a chain member, and that robots in the search state do not react when they perceive the prey.
- *Explore* → *Search*: if no chain member is perceived any more.
- *Explore* → *Chain*: if the tail of a chain is reached (i.e., only one chain member is perceived) and either (i) the prey is not perceived, in which case the robot joins the chain with probability P_{in} per time step; or (ii) the prey is perceived at a distance >30 cm.
- *Explore* → *Success*: 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 Vectorfield controller

The initial situation is the same as in the previous section. The robots first have to search for the nest. The first robot to find the nest stops moving and activates a colour pattern with its LEDs pointing towards the nest (see Fig. 2b). Another robot that perceives such a colour pattern will use the indicated direction to move away from the nest. It will end up joining 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. It activates its

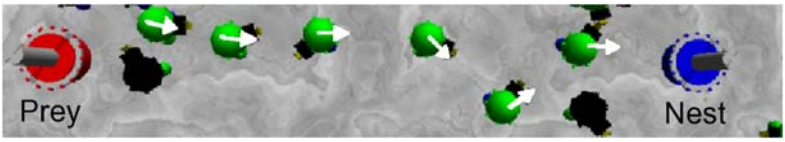


Fig. 5 A vectorfield. The six robots with an *arrow* on their top represent a path between nest and prey. The *arrows* indicate the directions the robots are pointing to and that lead towards the nest

LEDs to point towards the LED-activated robot it perceives. The resulting robot structure can be considered as a vectorfield globally leading to the nest. As can be seen in Fig. 1c, the structure of the vectorfield exhibits stronger branching than the chain controller (Fig. 1b), because the vectorfield 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 vectorfield is probabilistic. At each time step a robot leaves the vectorfield with probability P_{out}, but only if it is situated at the vectorfield's border. Similarly to the chains, the process of joining/leaving the vectorfield leads to a continuous exploration of the environment until the prey is found. If a robot of the vectorfield perceives the prey it sets its value of P_{out} to zero, 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 vectorfield.

When a robot leaves the vectorfield, it starts a random walk during which it does not react to the perception of the vectorfield. Due to the random walk it might reach a different branch, stay in the vicinity of the same branch, or lose contact with the vectorfield 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 vectorfield. This process is continued until the vectorfield encounters the prey 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 represent the directions they are pointing to. The state diagram of the vectorfield controller is given in Fig. 6. In addition to the motor schemas employed for the chain controller, one more motor schema is required for the vectorfield. This motor schema, the behaviours and the behaviour transitions are detailed in the following.

Motor schemas

– *Follow_vectorfield(indicated_directions)*: takes into account directions as indicated by all perceived robots in the vectorfield, and returns a unit vector that points in the opposite direction.

Behaviours

– *Search*: perform a random walk (until the vectorfield is perceived). LEDs are off. Active motor schemas: Move_straight, Random, Avoid_collisions.
– *Explore*: follow the vectorfield away from the nest. LEDs are off. Active motor schemas: Follow_vectorfield, Avoid_collisions.
– *Vectorfield*: do not move. The LEDs are used to activate a pattern which indicates the direction towards the precedent robot in the vectorfield. Active motor schemas: none.
– *Random*: perform a random walk (even if the vectorfield is perceived). LEDs are off. Active motor schemas: Move_straight, Avoid_collisions, Random.
– *Success*: the experiment is successfully finished. No action is performed.

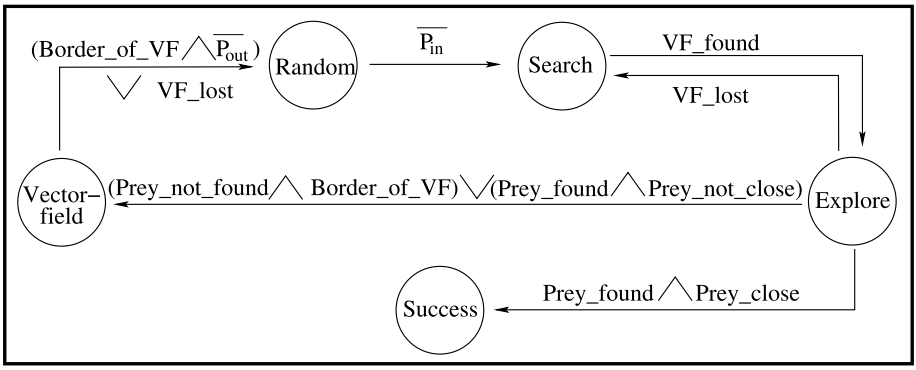


Fig. 6 State diagram of the vectorfield controller. Each *circle* represents a state (i.e., a behaviour). *Arrows* are labelled with the conditions that trigger a state transition. The initial state is the search state. The expressions $\overline{P_{\text{in}}}$ ($\overline{P_{\text{out}}}$) are Boolean variables that are set to *true* if $R \leq P_{\text{in}}$ ($R \leq P_{\text{out}}$), 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

Behaviour transitions

- *Search* → *Explore*: if the vectorfield is perceived. Note that the nest is perceived as part of the vectorfield, and that a robot searching for the nest does not react when it just perceives the prey.
- *Explore* → *Search*: if the vectorfield is no longer perceived.
- *Explore* → *Vectorfield*: (i) if the prey is not perceived and only one vectorfield robot is perceived at a distance >30 cm, or (ii) if the prey is perceived at a distance >30 cm.
- *Explore* → *Success*: if the prey is perceived at a distance <30 cm.
- *Vectorfield* → *Random*: (i) if the robot is situated at the border of the vectorfield, it leaves the vectorfield with probability P_{out} per time step, or (ii) if the vectorfield is no longer perceived.
- *Random* → *Search*: with probability P_{in} per time step.

4.3 Differences between chain and vectorfield controllers

There are three main differences between the chains and the vectorfield. 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 vectorfield 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 vectorfield as robots immediately join the vectorfield when they reach its border. This, in general, leads to a higher degree of branching of the vectorfield structure. Finally, the rule employed for leaving the vectorfield leads to a higher degree of randomness because a robot performs a random walk and might lose 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 the following, we specify the experimental setup, we briefly describe the methodology followed to determine good parameter sets for each controller, we present the experiments performed and discuss the results obtained.

5.1 Experimental setup

We employ a bounded arena of size 5 m × 5 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 (i.e., 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 to be a failure. As a second performance measure we use the *success rate* which we define as the ratio of successful trials.

5.2 Parameter settings and general performance

The overall behaviour of our controllers is a function of the two parameters P_{in} and P_{out}. The values of these determine the rates at which the robots join or leave the path forming structure. To assess the general impact of these two probabilities we have conducted a parameter study. For each probability we examined ten values defined by $0.001 \cdot 2^x$, with $x \in \{0, 1, 2, 3, \ldots, 9\}$, resulting in 100 candidates in the range $[0.001, 0.512]$. Figure 7 shows a surface plot of the success rate of the $P_{\text{in}}/P_{\text{out}}$ parameter landscape for a group of 40 robots in an environment without obstacles and a nest to prey distance of 3 meters.

The parameter landscapes are similar for the three chain variants. In general, the success rate is higher when $P_{\text{in}} > P_{\text{out}}$. Low values of P_{in} result in a rather patient behaviour: in most cases a single chain is formed slowly. For high values of P_{in}, several chains are formed fast and in parallel. The second parameter, P_{out}, determines the stability of the formed chains, directly influencing their lifetime and the frequency of chain disbandment. High values of P_{out} lead to an impatient behaviour where robots joining a chain quickly disaggregate from it. For a more detailed analysis of the chain parameter we refer the reader to Nouyan and Dorigo (2006).

For the vectorfield the most successful parameter combinations are in the proximity of the line $P_{\text{in}} = P_{\text{out}}$. If $P_{\text{in}} \ll P_{\text{out}}$ only few robots get aggregated into the forcefield, and if $P_{\text{in}} \gg P_{\text{out}}$ a structure is formed quickly, but the exploration of the environment is limited because the robots leave the vectorfield very rarely and then join it fast at a nearby position.

In general, the results are qualitatively similar for other group sizes, distances, and when obstacles are added to the environment. However, for the chains we found that for rather

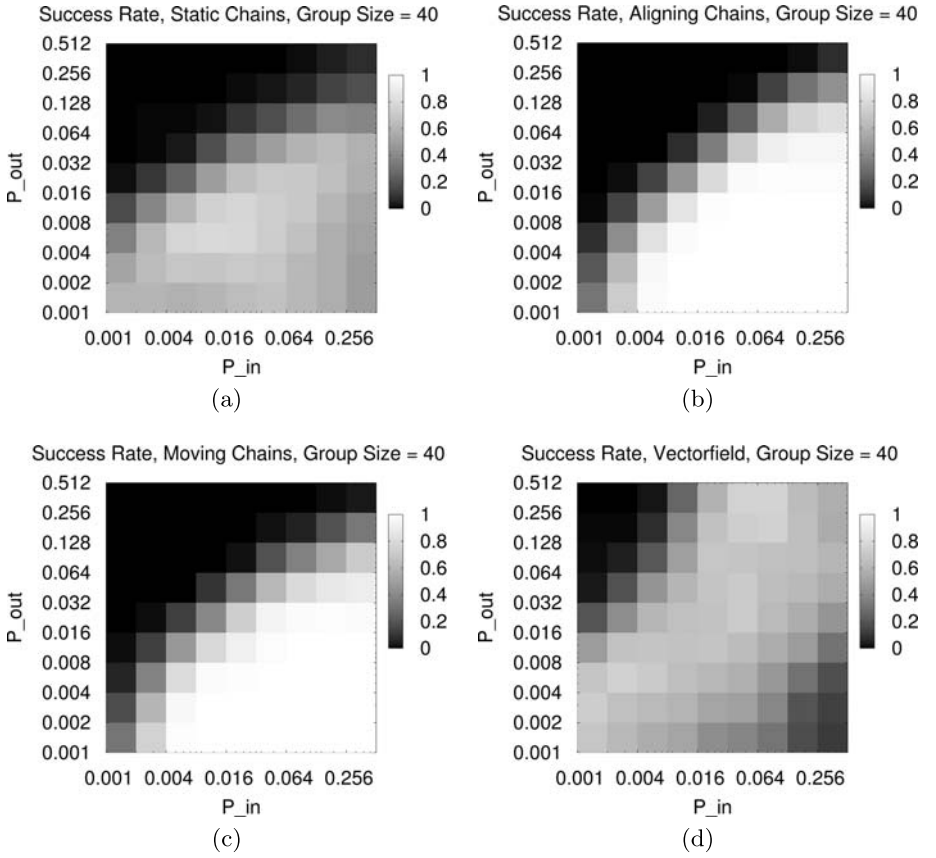


Fig. 7 Surface plot of the success rate of the parameter landscape when changing the two probability parameters P_{in} and P_{out} (100 observations per parameter combination), for a group of 40 robots in an environment without obstacles and a nest to prey distance of 3 meters, when applying **a** static chains, **b** aligning chains, **c** moving chains, or **d** the vectorfield. The axes of the parameters are plotted in logarithmic scale. The lighter the surface the higher the success rate

difficult setups with a small group size and a large distance between nest and prey, the highest success rates are reached for low values for the two probabilities. The robots then usually form a single chain, which is the only possibility to solve the task.

The study of the parameter landscapes gave a general idea of the impact of the parameters and of the overall performance of the controllers. To further analyse the performance under a wide range of experimental settings, as reported in Sect. 5.3, we have to select one parameter combination for each controller. For this purpose, we employed the racing method by Birattari (2002) and Birattari et al. (2005). 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}. We considered the same ten values defined by $0.001 \cdot 2^x$, with $x \in \{0, 1, 2, 3, \ldots, 9\}$,

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, s
Static chain	0.064	0.008	62.2	4142
Aligning chain	0.128	0.004	89.7	1066
Moving chain	0.128	0.004	91.9	1181
Vectorfield	0.064	0.016	48.1	>10000

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 $O \in \{0, 10, 20\}$. Each setup is initialized in 100 different ways, obtained varying the initial positions of the robots and the obstacle configuration. It is important to note that while for $N = 20$ and $N = 40$ a solution exists for any value of D and O, this is not necessarily the case for $N = 10$. In this case, in fact, the 10 robots can form a linear structure of approximately 3 m. Therefore, when $O = 0$ a solution exists, while when $O = 10$ or $O = 20$ the existence of the solution depends on the actual disposition of the obstacles in the arena.

For each controller there were between three and 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. The performance obtained with these candidates is reported in Table 1.

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 robots.

The vectorfield performs worse than the others, not forming a path in 51.9% of the cases. The lower success rate of the vectorfield and of the static chains is due to the following reasons. First, the structures they form do not move. 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 vectorfield, 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 vectorfield. 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 vectorfield or a chain when performing a random walk.

5.3 Experiments performed and results

We evaluated the four controllers with the parameters of Table 1, performing the following experiments:

- A difficulty test where we vary the distance between the nest and the prey in the range [1 m, 3 m].
- A scalability test where we vary the number of robots in the range [10, 200].
- An obstacle test where we vary the number of obstacles in the range [0, 30] and additionally test two predefined obstacle environments.
- A set of robustness tests where we vary the noise of various sensors.

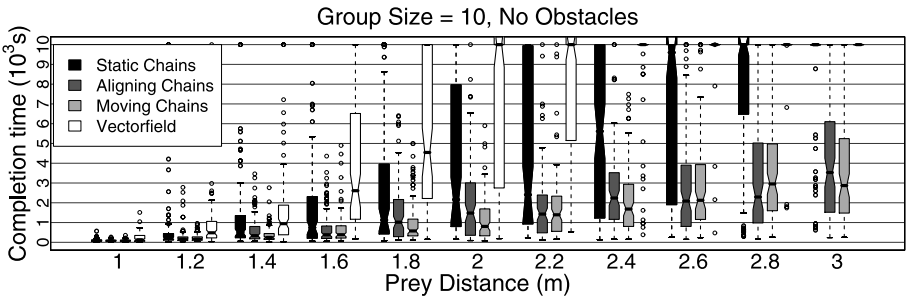


Fig. 8 The box-and-whisker-plot (Becker et al. 1988) shows 100 evaluations per box of the completion time when changing the nest to prey distance, for a group of 10 robots in an environment without obstacles. *Boxes* represent the inter-quartile range of the data, while the *horizontal bars* inside the *boxes* mark the median values. The whiskers extend to the most extreme data points within 1.5 of the inter-quartile range from the *box*. The *empty circles* mark the outliers

– A set of fault tolerance tests where we vary the fraction of robots that suffer from individual failure by disabling various sensors or actuators.

Difficulty test 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 Fig. 8.

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 Sect. 4, the results show that indeed for a small group size the vectorfield performs worse than the chain strategies. While the completion time of static chains and vectorfield increases quadratically with increasing distance to the prey, the completion time of two dynamic chain variants increases only linearly.

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 Fig. 9. First, Fig. 9a reports the completion times. The performance of all controllers, and in particular of the vectorfield, improves with the group size. While the vectorfield 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 bigger 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 vectorfield, 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, Fig. 9b displays the overall effort, that is, the product of completion time and robot group size. This measure is a good indicator of scalability and can be used to investigate super-linearity in the system. The usual way to do this is to divide—rather than multiply—performance by the robot group size. However, as in our case the specific performance metric of completion time has to be minimized (rather than maximized), a multiplication is required to obtain a normalized measure representing the cumulated effort of all robots. A decrease for growing group sizes means that the added resources lead to a more than

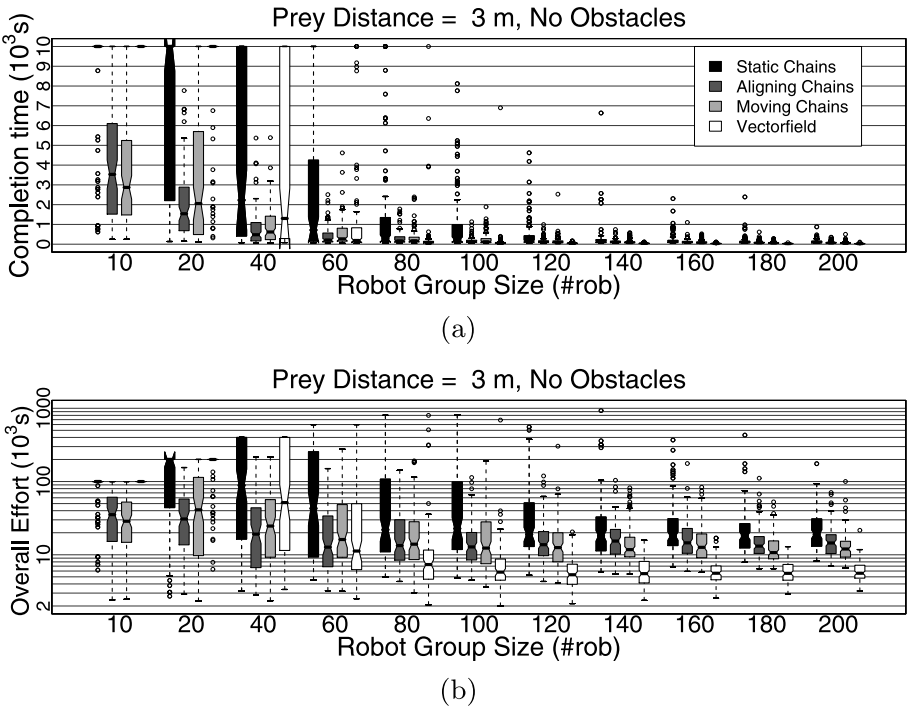


Fig. 9 Box-and-whisker plots (Becker et al. 1988) of the scalability test in an environment without obstacles (100 observations per box). **a** Completion time. **b** Overall effort (i.e., the product of completion time and group size); it measures the scalability of the system and is plotted in logarithmic scale. See the caption of Fig. 8 for an explanation of the box-and-whisker plots

proportional decrease in completion time. The results show that our controllers have good scalability: for all controllers the overall effort decreases up to 100 robots, and then remains roughly constant. In particular, the decrease in overall effort observed for vectorfield and static chains as group sizes grow shows a super-linear increase in efficiency. The observed super-linear effect is possible in our experimental conditions as there is no overcrowding, which would cause physical interferences between the robots. The performance of the static chains remains below the one of the dynamic chains throughout all group sizes.

Obstacle test To assess the performance in presence of obstacles, we tested our controllers in three types of obstacle environments. In addition to the standard arena with a random configuration of obstacle cubes (R-arena, Fig. 10a), we also used two predefined arenas with a fixed configuration of obstacles: the X-arena (Fig. 10b) and the U-arena (Fig. 10c).

Figure 11 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 in several ways. 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 increases the length of the shortest path, and implies a particular difficulty for the aligning and the moving strategies. In fact, these two strategies attempt to align the

Fig. 10 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, where the prey is positioned behind a U-shaped obstacle

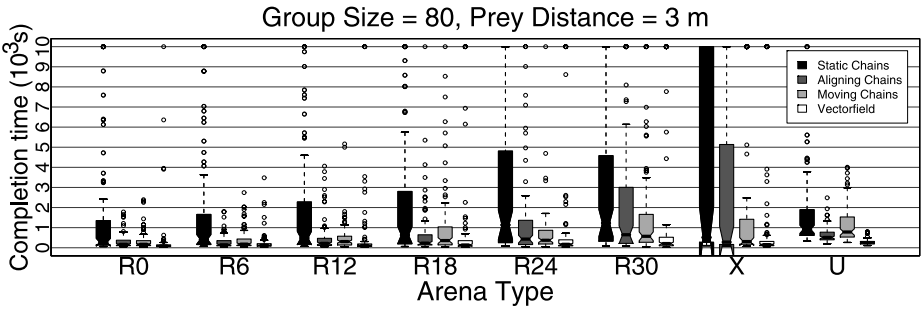


Fig. 11 Box-and-whisker plot (Becker et al. 1988) of the obstacle test (100 observations per box), for a group of 80 robots and a prey distance of 3 meters. For arenas of type R the number of obstacles is indicated. See the caption of Fig. 8 for an explanation of the box-and-whisker plots

chains. Due to this alignment, a chain can be broken up in cases where the line of sight of two neighboured chain members is blocked by an obstacle.

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, vectorfield 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, vectorfield performs considerably worse for smaller robot groups.

Robustness tests To analyse how the performance of our controllers changes in the presence of noisy conditions, we have conducted a series of robustness tests in which we vary the noise of the various sensors. The noise is calculated at each time step as a uniformly random value within the range $[-noise_{\max}; noise_{\max}]$, and is added to the considered sensor value. Figure 12 shows the results for a group size of 80 robots in an environment without obstacles and a nest to prey distance of 3 meters. We varied the noise of the direction to objects[3] perceived by the camera (Fig. 12a), the distance to objects perceived by the camera

[3] An object can be either another robot, the nest, or the prey. Obstacles cannot be perceived with the camera.

(Fig. 12b), the proximity sensor (Fig. 12c), and, only for the vectorfield, the direction-to-nest vector indicated by other robots in the vectorfield perceived by the camera (Fig. 12d).[4]

The results obtained show that the highest degree of sensitivity to sensor noise is observed for the camera direction. The performance of the vectorfield decreases linearly with growing level of noise. For the three chain variants the performance decreases with growing noise up to a noise level of 108°, and then remains roughly constant.

In the other three tests all controllers exhibit good robustness:

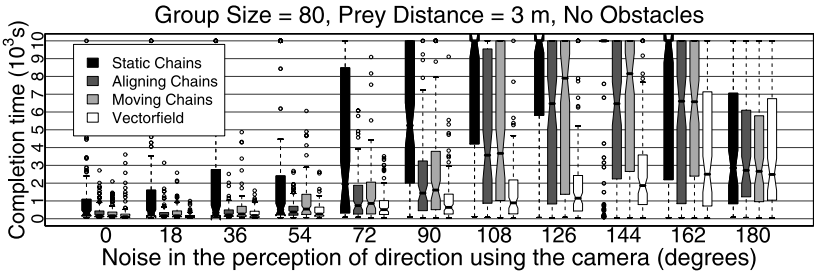
- For the perception of the distance to other objects using the camera, a high level of noise leads to a "nervous", oscillating behaviour as robots continuously try to adjust their distance.
- For the proximity sensors, a high level of noise leads to more collisions because obstacles are not correctly perceived. At the same time the robots may try to avoid non existent obstacles.
- For the perception of the direction indicated by robots in a vectorfield, a high level of noise leads to a higher degree of randomness of the vectorfield structure. However, the noise is effectively decreased if multiple robots are perceived. In this case all direction-to-nest vectors are taken into account and therefore the noise is reduced. As a result, even for a noise level of 180° the performance remains quite high.

Fault tolerance tests We conducted a series of fault tolerance tests to analyse how our controllers cope with individual failures. Figure 13 shows the results for a group of 80 robots in an environment without obstacles and a nest to prey distance of 3 meters. We disabled either the camera (Fig. 13a), the LED-ring (Fig. 13b), the proximity sensor (Fig. 13c), or the tracks (Fig. 13d) for a given fraction of the robots.

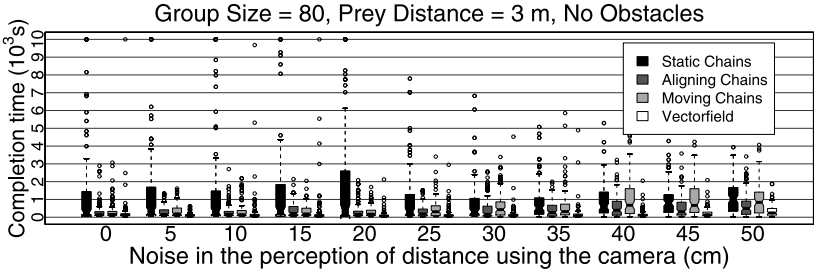
When a robot has no camera it just performs a random walk and can be considered as a mobile obstacle. It cannot join the path forming structure and therefore cannot contribute to the solution of the problem. Without LEDs robots are able to perceive the other robots, the nest, and the prey; and they try to join the path forming structure. However, as they cannot use their LEDs to signal their presence to the other robots they do not contribute to the solution of the problem either and can be considered as mobile obstacles. When the proximity sensors are disabled, a robot does not perceive any obstacles and may get stuck in a corner of the environment. However, if it perceives a part of the path forming structure it can join it. Therefore, a robot without proximity sensors can still contribute to forming a path. The system continues to perform, although with a lower performance level, even if the proximity sensors of all robots are disabled. With non-moving tracks a robot is not able to move and therefore can be considered as an immobile obstacle. It can, in principle, still join the path forming structure by chance if it happens to be positioned at the right place.

In general, for all four fault tolerance tests the performance decreases for increasing fractions of erroneous robots. However, up to an error rate of approximately 50% of the robots, the performance remains quite high in all considered cases. This can be explained by the large group size which leads to a high degree of redundancy, so that the remaining fully functional robots are still able to solve the task.

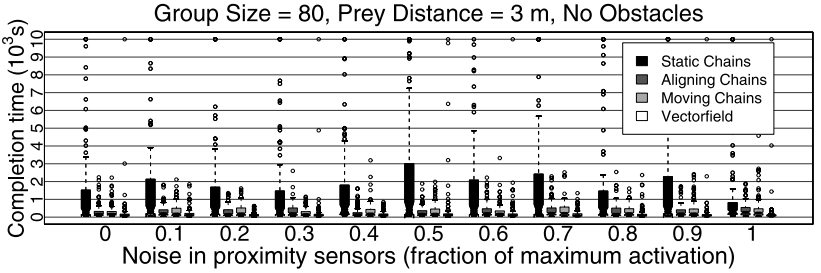
[4] As mentioned in Sect. 3, the values used in all experiments for these four noise levels are 18°, 10 cm, 0.2, and 36°. In the robustness tests we manipulate one value at a time.



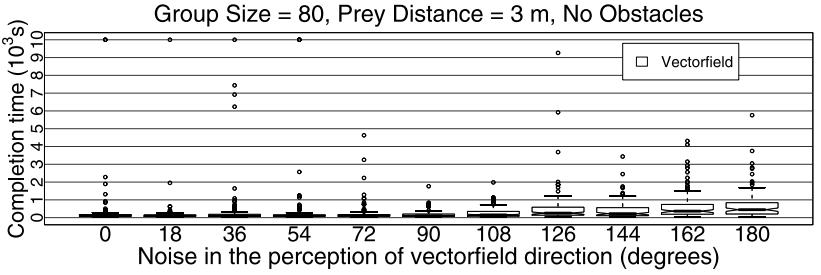
(a)



(b)

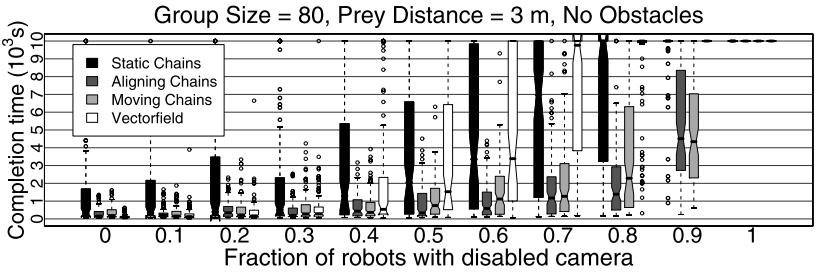


(c)

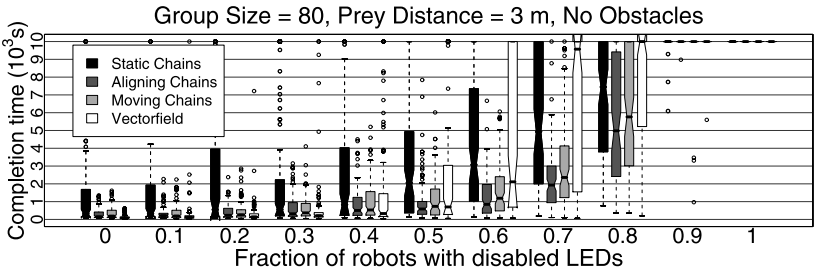


(d)

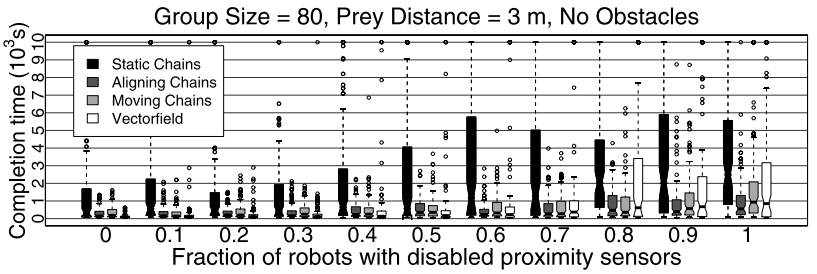
Fig. 12 Box-and-whisker plots (Becker et al. 1988) of the robustness tests for a group size of 80 robots in an environment without obstacles and a nest to prey distance of 3 meters (100 observations per box). We varied the noise of **a** the direction to objects perceived by the camera, **b** the distance to objects perceived by the camera, **c** the proximity sensor, and, only for the vectorfield, **d** the nest direction indicated by other robots in the vectorfield as perceived by the camera. See the caption of Fig. 8 for an explanation of the box-and-whisker plots



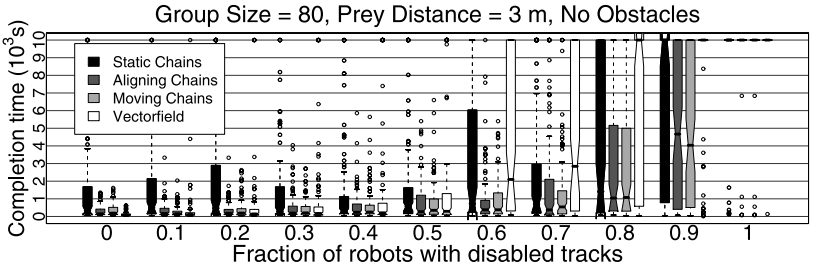
(a)



(b)



(c)



(d)

Fig. 13 Box-and-whisker plots (Becker et al. 1988) of the fault tolerance tests for a group size of 80 robots in an environment without obstacles and a nest to prey distance of 3 meters (100 observations per box). We varied the fraction of robots with disabled **a** camera, **b** LED-ring, **c** proximity sensor, and **d** tracks. See the caption of Fig. 8 for an explanation of the box-and-whisker plots

6 Related work

As mentioned in the introduction, traditional approaches to environment navigation are often based on an internal map-like representation of the environment (Filliat and Meyer 2003; Meyer and Filliat 2003). 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 (Mamei and Zambonelli 2005; Payton et al. 2001, 2004), or to ants (Svennebring and Koenig 2004).

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 (2004) use a sensor network with up to 156 Gnats sensor nodes that compute the shortest path using the distributed Bellman-Ford algorithm (Bellman 1957), and test the impact of different configurations of sensors placement. Li et al. (2003) 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 and Sukhatme (2002) 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 (2005) 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 (1992), 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. Similar systems were implemented by Drogoul and Ferber (1992), and by Cohen (1996). In the latter case the robot group is heterogeneous, such that there are two groups of robots, one taking care of path formation and the other exploiting the formed path to follow it to a goal location. All these 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. 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 vectorfield two colours for nest and prey, and one pattern for direction indication suffice.
Werger and Matarić (1996) 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. (2001, 2004) 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 vectorfield. 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 or by some rule.

7 Discussion and conclusions

We have presented an experimental study of two control mechanisms that employ visually connected robot structures to form a path between two objects. One of these mechanisms 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 mechanism, called vectorfield, leads to the formation of a tree-like structure with many branches. Both mechanisms are completely distributed and homogeneous, and make use of local information and communication only.

In an experimental study, we first studied the impact of the two probability parameters P_{in} and P_{out} that determine the rates at which the robots join or leave the path forming structure. Furthermore, we used a racing method to choose one good parameter set for each controller. We then extensively tested each controller under various conditions. In general, our controllers reached a good performance, also in the presence of obstacles. The chains perform better for groups of up to 40 robots, while the vectorfield 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.

As stated in the introduction, the main benefits that one seeks when pursuing a swarm robotics approach are (i) scalability with the number of robots, (ii) robustness with respect to noisy conditions, and (iii) fault tolerance in case of individual failure. We showed that for all our controllers these three characteristics were fulfilled.

We believe that the potentially higher degree of scalability of the vectorfield controller is due to its use of simpler rules and more randomness. The chain controllers are a slightly more engineered solution, by which we tried to obtain 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. In this way the robots avoid getting lost. On the other hand, when leaving a vectorfield, a robot starts a random walk with the risk of losing contact with the vectorfield. While this may be a disadvantage for small group sizes, it turns out to work quite well for larger robot groups, where the chain mechanisms have the disadvantage of multiple robots trying to follow the same path which may lead to a congestion in the path.

An interesting extension to our control strategies would be a mechanism that allows the formation of paths from both the nest and the prey. Although such a mechanism could significantly speed up the path formation process, it would also add complexity to both the control algorithm and to the communication protocol among the robots, as additional signals would be necessary to distinguish between the two different types of paths.

One of the main lessons we learned is that for a low density of robots it is beneficial to use linear robot structures to form a path, such as our chains. When the density of robots is high, a more branched robot structure such as the vectorfield is a better choice. A general guideline could be to start with the vectorfield control mechanism, and then to switch to chains if the prey is not found after some time, as this indicates a low density of robots. The robots could communicate their desire to switch the control algorithm, for instance, by using their speakers and microphones, and in this way trigger a global switch. Another interesting possibility would be to use chains and vectorfield at the same time. Robots could then individually decide which controller to use based on their local perception, and different areas of the environment might be explored by groups of robots aggregating into chains or into a forcefield.

In the future we intend to implement our control mechanisms on 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 cannot be moved by a single robot (Nouyan et al. 2006). The vectorfield 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 vectorfield (Campo et al. 2006).

Acknowledgements 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 "SWARMANOID Project", funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888. 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. The authors thank Christos Ampatzis, Mauro Birattari, Anders Lyhne Christensen, Roderich Groß, Thomas Halva Labella, Francisco Santos, and Vito Trianni for stimulating discussions and for their feedback during the preparation of this paper. The authors also thank the editor Alcherio Martinoli and the anonymous reviewers for their feedback. Alexandre Campo and Marco Dorigo acknowledge support from the fund for scientific research F.R.S.—FNRS of Belgium's French Community of which they are respectively Research Fellow and Research Director.

References

Arkin, R. C. (1998). *Behavior-based robotics*. Cambridge: MIT Press.

Batalin, M., & Sukhatme, G. S. (2002). Spreading out: a local approach to multi-robot coverage. In *Proceedings of the sixth international symposium on distributed autonomous robotic systems* (pp. 373–382). Berlin: Springer.

Becker, R. A., Chambers, J. M., & Wilks, A. R. (1988). *The new S language. A programming environment for data analysis and graphics*. London: Chapman and Hall.

Bellman, R. E. (1957). *Dynamic programming*. Princeton: Princeton University Press.

Birattari, M. (2005). *The problem of tuning metaheuristics as seen from a machine learning perspective*. *DISKI* (Vol. *292*). Berlin: AKA/IOS Press.

Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In W. B. Langdon et al. (Eds.), *Proceedings of the genetic and evolutionary computation conference* (pp. 11–18). San Francisco: Morgan Kaufmann.

Campo, A., Nouyan, S., Birattari, M., Groß, R., & Dorigo, M. (2006). Negotiation of goal direction for cooperative transport. In M. Dorigo et al. (Eds.), *LNCS*: *Vol. 4150. Ant colony optimization and swarm intelligence: 5th international workshop, ANTS 2006* (pp. 191–202). Berlin: Springer.

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

Christensen, A. L., & Dorigo, M. (2006). Evolving an integrated phototaxis and hole-avoidance behavior for a swarm-bot. In *Artificial life X: proceedings of the tenth international conference on the simulation and synthesis of living systems* (pp. 248–254). Cambridge: MIT Press.

Christensen, A. L., O'Grady, R., & Dorigo, M. (2007, in press). Morphology control in a self-assembling multi-robot system. *IEEE Robotics & Automation Magazine*.

Christensen, A. L., O'Grady, R., Birattari, M., & Dorigo, M. (2008, in press). Fault detection in autonomous robots based on fault injection and learning. *Autonomous Robots*.

Cohen, W. W. (1996). Adaptive mapping and navigation by teams of simple robots. *Robotics and Autonomous Systems*, *18*, 411–434.

Deneubourg, J.-L., Aron, S., Goss, S., & Pasteels, J.-M. (1990). The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*, *3*, 159–168.

Drogoul, A., & Ferber, J. (1992). 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)* (pp. 451–459). Cambridge: MIT Press.

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

Goss, S., & Deneubourg, J.-L. (1992). Harvesting by a group of robots. In *Proc. of the 1st European conf. on artificial life* (pp. 195–204). Cambridge: MIT Press.

Howard, A. (2004). Multi-robot mapping using manifold representations. In *Proc. of the 2004 IEEE int. conf. on robotics and automation* (pp. 4198–4203). Los Alamitos: IEEE Computer Society.

Jakobi, N. (1997). Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior*, *6*, 325–368.

Jakobi, N., Husbands, P., & Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In F. Morán, A. Moreno, J. J. Merelo, & P. Chacón (Eds.), *Lecture notes in artificial intelligence*: *Vol. 929*. *Advances in artificial life. Proceedings of the 3rd European conference on artificial life (ECAL 1995)* (pp. 704–720). Berlin: Springer.

Li, Q., De Rosa, M., & Rus, D. (2003). Distributed algorithms for guiding navigation across a sensor network. In *9th international conference on mobile computing and networking* (pp. 313–325). New York: ACM.

Mamei, M., & Zambonelli, F. (2005). Physical deployment of digital pheromones through RFID technology. In *Proc. of the 4th int. conf. on autonomous agents and multi-agent systems, AAMAS 2005* (pp. 1353–1360). IEEE Press: Piscataway.

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

Miglino, O., Lund, H. H., & Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life*, *4*, 417–434.

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

Nouyan, S., & Dorigo, M. (2006). Chain based path formation in swarms of robots. In M. Dorigo et al. (Eds.), *LNCS*: *Vol. 4150*. *Ant colony optimization and swarm intelligence: 5th international workshop, ANTS 2006* (pp. 120–131). Berlin: Springer.

Nouyan, S., Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2006). Group transport along a robot chain in a self-organised robot colony. In *Proc. of the 9th int. conf. on intelligent autonomous systems* (pp. 433–442). Amsterdam: IOS Press.

O'Hara, K. J., & Balch, T. R. (2004). Pervasive sensor-less networks for cooperative multi-robot tasks. In *Proceedings of the seventh international symposium on distributed autonomous robotic systems*. Berlin: Springer.

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

Payton, D., Estkowski, R., & Howard, M. (2004). Pheromone robotics and the logic of virtual pheromones. In E. Şahin et al. (Ed.), *LNCS*: *Vol. 3342*. *Swarm robotics: SAB 2004 international workshop* (pp. 45–57). Berlin: Springer.

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

Werger, B., & Matarić, M. (1996). 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)* (pp. 625–634). Cambridge: MIT Press.