Distributed Algorithms for Autonomous Mobile Robots

Giuseppe Prencipe[1] and Nicola Santoro[2]

[1] Dipartimento di Informatica, Università di Pisa, prencipe@di.unipi.it
[2] School of Computer Science, Carleton University, santoro@scs.carleton.ca

Abstract. The distributed coordination and control of a team of autonomous mobile robots is a problem widely studied in a variety of fields, such as engineering, artificial intelligence, artificial life, robotics. Generally, in these areas, the problem is studied mostly from an empirical point of view. Recently, a significant research effort has been and continues to be spent on understanding the fundamental algorithmic limitations on what a set of autonomous mobile robots can achieve. In particular, the focus is to identify the minimal robot capabilities (sensorial, motorial, computational) that allow a problem to be solvable and a task to be performed. In this paper we describe the current investigations on the interplay between robots capabilities, computability, and algorithmic solutions of coordination problems by autonomous mobile robots.

1 Introduction

In this paper we describe the current investigations on the algorithmic limitations of what autonomous mobile robots can do with respect to basic coordination problems.

The current trend in robotic research, both from engineering and behavioral viewpoints, has been to move away from the design and deployment of few, rather complex, usually expensive, application-specific robots. In fact, the interest has shifted towards the design and use of a large number of "generic" robots which are very simple, with very limited capabilities and, thus, relatively inexpensive, but capable, together, of performing rather complex tasks. The advantages of such an approach are clear and many, including: reduced costs (due to simpler engineering and construction costs, faster computation, development and deployment time, etc); ease of system expandability (just add a few more robots) which in turns allows for incremental and on-demand deployment (use only as few robots as you need and when you need them); simple and affordable fault-tolerance capabilities (replace just the faulty robots); re-usability of the robots in different applications (reprogram the system to perform a different task). Moreover, tasks that could not be performed at all by a single agent become manageable when many simple units are used instead [19, 34].

One of the first studies conducted in this direction in the AI community is that of Matarić [30]. The main idea in Matarić's work is that "interactions

Please use the following format when citing this chapter:

Prencipe, G., Santoro, N., 2006, in International Federation for Information Processing, Volume 209, Fourth IFIP International Conference on Theoretical Computer Science-TCS 2006, eds. Navarro, G., Bertossi, L., Kohayakwa, Y., (Boston: Springer), pp. 47–62.

between individual agents need not to be complex to produce complex global consequences".

Other investigations in the AI community include the study of [4] on stigmergy communication and on the use a set of simple robots that operate completely autonomously and independently to collect pucks spread over a square arena in a single cluster; the ALLIANCE architecture and the studies on selfish behavior of cooperative robots in animal societies by Parker [34]; the formation and navigation problems in multi-robot teams in the context of primitive animal behavior in pattern formation by Balch and Arkin [3]; and the experiments in cooperative cleaning behavior of Jung *et al* [28].

Alternative approaches to the problem of studying multi-robot systems, can be found in the CEBOT system of Fucuda, Kawaguchi *et al.* [25, 29], in the planner-based architecture of Noreils [32], in the information requirements theory of Donald *et al.* [19] (see [6] for a survey), in the Swarm Intelligence of Beni and Hackwood [5], in the Self-Assembly Machine ("fructum") of Murata *et al.* [31], etc.

The common feature of all these approaches is that they do not deal with formal correctness and they are only analyzed empirically. In all these investigations, algorithmic aspects were somehow implicitly an issue, but clearly not a major concern, let alone the focus, of the study. An investigation with an algorithmic flavor has been undertaken within the AI community by Durfee [20], who argues in favor of limiting the knowledge that an intelligent robot must possess in order to be able to coordinate its behavior with others.

Recently, the problem has been tackled from a different perspective: from a *computational* point of view. In other words, the focus is to understand the relationship between the capabilities of the robots and the solvability of the tasks they are given. In these studies, the impact of the *knowledge* of the environment is analyzed: can the robots form an arbitrary geometric pattern if they have a *compass*? Can they gather in a point? Which information each robot must have about its fellows in order for them to collectively achieve their goal? The goal is to look for the minimum power to give to the robots so that they can solve a given task; hence, to formally analyze the strengths and weaknesses of the distributed coordination and control.

In this paper we describe the current investigations on the interplay between robots capabilities, computability, and algorithmic solutions of coordination problems by autonomous mobile robots.

In Section 2 we describe the model used in these investigations. In Section 3 we review some results related to the analysis of the problem of pattern formation by autonomous mobile robots. Finally, in Section 4 we draw some conclusions and present suggestions for further study.

2 Modeling Autonomous Mobile Robots

In the general model, the computational universe is a 2-dimensional plane populated by a set of n autonomous mobile robots, denoted by $r_1, \ldots, r_n$, that are modeled as devices with computational capabilities which are able to freely move on a two-dimensional plane.

2.1 The robots and their behavior

A robot is a computational unit capable of sensing the positions of other robots in its surrounding, performing local computations on the sensed data, and moving towards the computed destination. The local computation is done according to a deterministic algorithm that takes in input the sensed data (i.e., the robots' positions), and returns a destination point towards which the executing robot moves. All the robots execute the same algorithm. The local view of each robot includes a unit of length, an origin, and a Cartesian coordinate system defined by the *directions* of two coordinate axes, identified as the x and y axis, together with their *orientations*, identified as the positive and negative sides of the axes.

Each robot repeatedly cycles through four *states*: (i) initially it is inactive – *Wait*, (ii) it observes the positions of the other robots in its area of visibility – *Look*, (iii) it computes its next destination point by executing the algorithm (the same for all robots) – *Compute*, and (iv) it moves towards the point it just computed – *Move*. After the *Move* it goes back to the *Wait* state.

The sequence: *Wait* - *Look* - *Compute* - *Move* form a *computation cycle* (or briefly *cycle*) of a robot. The operations performed by each robot r in each state will be now described in more details.

1. **Wait.** The robot is idle. A robot cannot stay indefinitely idle. Initially, all robots are in *Wait*.
2. **Look.** The robot observes the world by activating its sensors which will return a *snapshot* of the positions of all other robots within the visibility range with respect to its local coordinate system. Each robot is viewed as a point, hence its position in the plane is given by its coordinates, and the result of the snapshot (hence, of the observation) is just a set of coordinates in its local coordinate system: this set forms the *view of the world* of r.
3. **Compute.** The robot performs a *local computation* according to a deterministic algorithm $\mathcal{A}$ (we also say that the robot *executes* $\mathcal{A}$). The algorithm is the same for all robots, and the result of the *Compute* state is a *destination point*.
4. **Move.** If the destination point is the current location of r, r performs a *null movement* (i.e., it does not move); otherwise it moves towards the computed destination but it can stop anytime during its movement[1].

[1] e.g. because of limits to the robot's motorial capabilities.

The robots are completely *autonomous*: no central control is needed. Furthermore they are *anonymous*, meaning that they are a priori indistinguishable by their appearance, and they do not (need to) have any kind of identifiers that can be used during the computation.

Moreover, there are no explicit direct means of communication: any communication occurs in a totally implicit manner. Specifically, it happens by means of observing the robots' positions in the plane, and taking a deterministic decision accordingly. In other words, the only mean for a robot to send information to some other robot is to move and let the others observe (reminiscent of bees in a bee dance).

2.2 Levels of Synchronization

The model, in its general setting, makes no assumptions about the level of synchronization of the robots. Indeed, the assumptions on the level of synchronization have a deep impact on computability; in fact, there are problems that are unsolvable in the general setting but can be solved in a synchronous setting (e.g., see [36]). The situation is analogous to the one occurring in the distributed computing field, and the settings will be described in this section.

General Setting: Asynchronous Robots In the general setting, no assumptions on the cycle time of each robot, and on the time each robot takes to execute each state of a given cycle are made. It is only assumed that each cycle is completed in finite time, and that the distance traveled in a cycle is finite. Moreover, the robots do not need to have a common notion of time, and each robot can execute its actions at unpredictable time instants.

More precisely, there are only two limiting assumptions. The first one refers to space; namely, the distance traveled by a robot during a computational cycle.

Assumption A1 (Finite Distance). *The distance traveled by a robot r in a move is not infinite. Furthermore, there exists an arbitrarily small constant $\delta_r > 0$, such that if the destination point is closer than δ_r, r will reach it; otherwise, r will move towards it of at least δ_r.*

As no other assumptions on space exist, the distance traveled by a robot in a cycle is unpredictable.

The second limiting assumption is on the length of a cycle.

Assumption A2 (Finite Cycle). *The amount of time required by a robot r to complete a computational cycle is not infinite. Furthermore, there exists a constant $\varepsilon_r > 0$ such that the cycle will require at least ε_r time.*

As no other assumption on time exists, the resulting system is *fully asynchronous* and the duration of each activity (or inactivity) is unpredictable.

There are two important consequences:

1. Since the time that passes after a robot starts observing the positions of all others and before it starts moving is arbitrary, but finite, the actual move of a robot may be based on a situation that was observed arbitrarily far in the past, and therefore it may be totally different from the current situation.
2. Since movements can take a finite but unpredictable amount of time, and different robots might be in different states of their cycles at a given time instant, it is possible that a robot can be seen *while* it is moving by other robots that are observing[2].

These consequences render difficult the design of an algorithm to control and coordinate the robots. For example, when a robot starts a *Move*, it is possible that the movement it performs is not "coherent" with the current configuration (i.e., the configuration it observed at the time of the *Look* and the configuration at the time of the *Move* can differ), since, during the *Compute*, other robots can have moved.

Restricted Setting: Semi-synchronous Robots A computational setting that has been extensively investigated is one in which the cycles of all the robots are synchronized and their actions are atomic.

In particular, there is a global clock tick reaching all robots simultaneously, and a robot's cycle is an instantaneous event that starts at a clock tick and ends by the next.

The only unpredictability (hence the name *semi-synchronous*) is given by the fact that at each clock tick, every robot is either *active* or *inactive*, and only active robots perform their cycle. The unpredictability is restricted by the fact that at least one robot is active at every time instant, and every robot becomes active at infinitely many unpredictable time instants. A very special case is when every robot is active at every clock tick; in this case the robots are *fully synchronized.*

In this setting, at any given time, all active robots are executing the same cycle state; thus no robot will look while another is moving. In other words, a robot observes other robots only when they are stationary. This implies that the computation is always performed based on accurate information about the current configuration.

Furthermore, since no robot can be seen *while* it is moving, the movement can be considered *instantaneous.*

An additional consequence of atomicity and synchronization is that, for them to hold, the maximum distance that a robot can move in one cycle is bounded.

2.3 Capabilities

Different settings arise from different assumptions that are made on the robots' capabilities, and on the amount of information that they share and use during the accomplishment of the assigned task. In particular,

[2] Note that this does not mean that the observing robot can distinguish a moving robot from a non moving one.

- **Visibility.** The robots may be able to sense the complete plane or just a portion of it. We will refer to the first case as the *Unlimited Visibility* case. In contrast, if each robot can sense only up to a distance $V > 0$ from it, we are in the *Limited Visibility* case. In the following, we will say also that the robots have unlimited/limited visibility.
  In addition, a robot cannot in general detect whether there is more than one fellow robot on any of the observed points, included the position where the observing robot is. We say it cannot detect *multiplicity.*

- **Agreement on Coordinate System.** The robots do not necessarily share the same $x-y$ coordinate system, and do not necessarily agree on the location of the origin (that we can assume, without loss of generality, to be placed in the current position of the robot), or on the unit distance. In general, there is no agreement among the robots on the chirality of the local coordinate systems (i.e., in general they do not share the same concept of where North, East, South, and West are). We will refer to this scenario as *no agreement* on the local coordinate systems. In the most favorable scenario, the robots agree on the direction and orientation of both axes. In this case, we will talk of *total agreement* on the local coordinate systems. Note that knowledge of the directions and orientations of both axes does not imply knowledge of the origin or the unit of length. An intermediate scenario is when the robots agree only on the direction and orientation of one axis; we will talk of *partial agreement.*

- **Memory.** The robots can access local memory to store different amount of information regarding the positions in the plane of their fellows. In particular, if the robots can only store the robots' positions retrieved in the current observation, we have *oblivious* robots. In contrast, if the robots can store all the positions retrieved since the beginning of the computation, we have *unbounded memory* robots. We will also refer to the algorithm the robots execute as *oblivious* or *non oblivious*, depending on the assumption made.

Note that, the conditions under which the robots operate are by definition common knowledge among the robots.

Let us stress that the only means for the robots to coordinate is the observation of the others' positions and their change through time. For oblivious robots, even this form of communication is impossible, since there is no memory of previous positions.

3 Problems and Limitations

In the following, we survey the computational results obtained so far. They are mostly about *geometric* problems, like forming a certain pattern, following a trail, or deploy the robots in order to guarantee optimal coverage of a certain terrain. Observe that several classical problems in distributed computing (e.g.,

leader election) can be reformulated as geometric problems in our model (e.g., forming an asymmetric pattern).

3.1 Pattern formation

The PATTERN FORMATION problem is one of the most important coordination problem and has been extensively investigated in the literature (e.g., see [8, 38, 39, 41]). The problem is practically important, because, if the robots can form a given pattern, they can agree on their respective roles in a subsequent, coordinated action. The geometric pattern to be formed is a set of points (given by their Cartesian coordinates) in the plane, and it is initially known by all the robots in the system.

The robots are said to *form the pattern* if, at the end of the computation, the positions of the robots coincide, in everybody's local view, with the points of the pattern. The formed pattern may be *translated*, *rotated*, *scaled*, and *flipped* into its mirror position with respect to the initial pattern. Initially the robots are in arbitrary positions, with the only requirement that no two robots are in the same position, and that, of course, the number of points prescribed in the pattern and the number of robots are the same.

The basic research questions are which patterns can be formed, and how they can be formed. Many proposed procedures do not terminate and never form the desired pattern: the robots just converge towards it; such procedures are said to *converge.*

Arbitrary Pattern In this section, we review our results on the formation of an arbitrary pattern. The problem has been investigated by Flocchini *et al* [21, 23] and Oasa *et al.* [33] in the general setting, and by Suzuki and Yamashita [39] in the semi-synchronous setting; both investigations consider robots with unlimited visibility.

In the *general setting* with *unlimited visibility*:

- With total agreement *oblivious* robots can form any arbitrary given pattern [21].
- With partial agreement, *oblivious* robots can form any arbitrary given pattern if n is odd. If n is even, *oblivious* robots can form only symmetric patterns that have at least one axis of symmetry not passing through any vertex of the pattern [23].
- With no agreement at all, *oblivious* robots cannot form an arbitrary given pattern [21].

In the *semi-synchronous setting* with *unlimited visibility*, let m be the size of the largest subset of robots having an equivalent initial view.

- Robots with *unbounded memory* can form [39]
  1. any pattern if $m = 1$;
  2. only patterns whose vertices can be partitioned into n/m regular m-gons all having the same center, if $m \geq 2$.

Circle Formation In the CIRCLE FORMATION problem, the robots want to place themselves on the plane to form a non degenerated circle (i.e., with finite radius greater than zero).

First observe that, if the diameter of the circle is not fixed a priori, the problem can be solved in a rather straightforward way by *oblivious* robots even in the *general setting*: each robot computes the smallest circle enclosing all the robots' positions and moves on the circumference of such a circle.

The problem becomes more difficult when the diameter is prescribed. This problem was first studied by Sugihara and Suzuki [38]. They presented an heuristic distributed protocol that allowed the robots to form an approximation of a circle having a given diameter. The distributed protocol they proposed (executed independently by all the robots) to let the robots form an approximation of a circle of given diameter D. Experiments have shown that sometimes the robots bring themselves in a configuration similar to a Reuleaux triangle rather than a circle (see Figure 1). Successively, the protocol has been improved by Tanaka [40], that proposed a new solution that produces a better approximation of the circle.

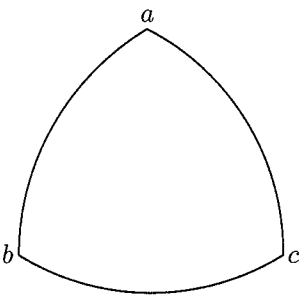

Fig. 1. Reuleaux's triangle. It is obtained by drawings arcs $arc(a, b)$, $arc(b, c)$, and $arc(c, a)$, with radii equal to D, from the vertices c, a, and b, respectively, of an equilater triangle $\triangle(a, b, c)$ with sides equal to D.

A variant of this problem is the UNIFORM CIRCLE FORMATION problem: the n robots on the plane must be arranged at regular intervals on the boundary of a circle. Notice that this is the same as the problem of forming an n-gon. This problem has been studied in the semi-synchronous setting by Défago and Konagaya [16]; simulation results of these studies have been presented in [37].

The solution in [16] is, however, computationally expensive: in fact, it involves the use of Voronoi diagrams, necessary to avoid the very specific possibility in which at least two robots share at some time the same position and also have total agreement. Based on this observation, in [7] it is presented a new algorithm that avoids these expensive calculations.

- In the *semi-synchronous setting* with *unlimited visibility*: *oblivious* robots can converge towards an n-gon [16, 37, 7].

Line Formation Let us now consider another simple pattern for the robots: a line. That is, the robots are required to place themselves on a line, whose position is not prescribed in advance; we just defined the LINE FORMATION problem. Note that, if $n = 2$, a line is always formed. Despite the simplicity of its formulation, this problem has some subtleties that render its solution not so easy. In fact, the solvability of this problem heavily depends on the amount of agreement the robots have on their local coordinate systems.

Clearly, if the robots can rely on *total agreement*, then the problem is easily solved: after lexicographically ordering the robots' positions (e.g., left-right, top-down), the first and the last robot in the ordering define the line to be formed. Then, all robots move sequentially (in order to avoid collisions) to this line (see Figure 2.a).

If the robots have *partial agreement*, for instance on the direction and orientation of y, the robots can not rely on an unique total ordering of the robots' positions. In this case the robots can place themselves on the axis that is median between the two vertical axes tangent to the observed configuration (see Figure 2.b). The robots on the tangent axes are the last to move.

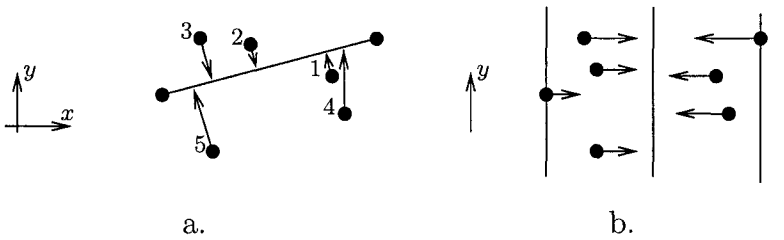

Fig. 2. Line formation with (a) total and (b) partial agreement.

In a recent study [15], the LINE FORMATION problem has been tackled by studying an apparently totally different problem: the *spreading*. In this problem, the robots, that at the beginning are arbitrarily placed on the plane, are required to evenly spread within the perimeter of a given region. In their work, the authors focus on the one-dimensional case: in this case, the robots have to form a line, and place themselves uniformly on it. A very interesting aspect of the study, is that [15] addresses the issue of *local algorithms*: each robots decides where to move based on the positions of its close neighbors. In particular, in the case of the line, the protocol, called *Spread*, is quite simple: each robot r observes its left and right neighbor. If r does not see any robot, it simply does not move; otherwise, it moves to the median point between its two neighbors. The authors prove its convergence in the semi-synchronous setting.

Semi-synchronous	Multiplicity Detection [39] Infinite Time [2, 13, 14]
Asynchronous	Multiplicity Detection [10] Compass [22] Unbounded Memory [9] Infinite Time [12]

Table 1. Summary of additional assumptions made by the existing solutions for the GATHERING problem.

- In the semy-synchronous setting, the robots executing *Spread* converge to a line configuration with equal distances.

Furthermore, if each robot knows the exact number of robots at each of its sides, it is possible to achieve the spreading in one dimension in a finite number of cycles.

- In the fully-synchronous model, n robots can spread in one dimension in $n-2$ cycles.

3.2 Gathering

In the GATHERING problem, the robots, initially placed in arbitrary positions, are required to gather in a single point. This problem is also called *point formation*, *homing*, or *rendezvous*.

In spite of its apparent simplicity, it has recently been tackled by several studies: in fact, several factors render this problem difficult to solve, as shown by the following

- In both the asynchronous and the semi-synchronous setting, there exists no deterministic oblivious algorithm that solves the GATHERING problem in a finite number of cycles, hence in finite time, for a set of $n \geq 2$ robots [35].

Some additional capabilities are thus needed to solve this problem (in Table 1 we report the existing results related to the GATHERING problem).

Let us first consider the case of *unlimited* visibility.

- In the semi-synchronous setting, $n \geq 3$ *oblivious* robots with *multiplicity detection* can gather in finite time [39].

This result has been recently improved; in fact, the same result can be achieved even in the general setting, extending the previous work of [11]:

- In the *general setting*, $n \geq 3$ *oblivious* robots with *multiplicity detection* can gather in finite time [10].

The *multiplicity detection* assumption is crucial to prove the correctness of these algorithms. In fact, the main idea is first to create a unique point p on the plane with two robots on it, and then to move all other robots on this point, taking care in not having other points with multiplicity greater than one while the robots move towards p.

In contrast, the multiplicity detection is not used in the solution described in [9]; however, it is assumed that the robots can rely on an unlimited amount of memory: the robots are said to be *non-oblivious*. In other words, the robots have the capability to store the results of all computations since the beginning, and freely access to these data and use them for future computations.

- In the *general setting*, $n \geq 3$ robots *with unbounded memory* can gather in finite time [9].

Another study [13] has been devoted to study the behavior of a particular simple solution to the problem: the robots use the center of gravity as gathering destination. The authors prove that this simple algorithm represent a convergence solution to the problem in the semi-synchronous setting. In [12] the same algorithm has been proven to be a convergence solution to the problem in the asynchronous setting.

Let us then consider the case of *limited* visibility. With limited visibility, an obvious necessary condition to solve the problem, is that at the beginning of the computation the *visibility graph* (having the robots as nodes and an edge (r_i, r_j) if r_i and r_j are within viewing distance) is connected [2, 22]. In [2] the proposed protocol works in the semi-synchronous setting; however, it is a *convergence* solution to the problem: the robots do not gather in finite time. In fact, the authors design a protocol that guarantees only that the robots converge towards the gathering point. In contrast, in [22], the authors present an algorithm that let the robots to gather in a finite number of cycles. However, the robots can rely on the presence of a common coordinate system: that is, they share a compass.

- In the semi-synchronous setting there exists an *oblivious* procedure that lets robots *converge* towards (but not necessarily reach) a point for any n [2].
- In the *general setting oblivious* robots with agreement on the coordinate system (e.g., with a compass) can gather in finite time [22, 24].

The GATHERING problem has been also investigated in the context of robots *failures*. In this context, the goal is for the non-faulty robots to gather regardless of the action taken by the faulty ones. Two types of robot faults were investigated by Peleg *et al.* [1]: *crash* failure, in which the robot stops any activity and will no longer execute any computational cycle; and the *byzantine* failure, in which the robot acts arbitrarily and possibly maliciously.

- In the semi-synchronous setting, gathering with at most one *crash* failure is possible [1].
- In the semi-synchronous setting, gathering with at most one *byzantine* failure is *impossible* [1].

- In the fully synchronous setting, gathering with at most $\frac{n-1}{3}$ *byzantine* failure is *possible* [1].

Finally, in [14] it is analyzed the case of systems where the robots have inaccuracies in sensing the positions of other robots, in computing the next destination point, and in moving towards the computed destination. The authors provide a set of limitations on the amount of inaccuracies allowing convergence; hence, they present an algorithm for convergence under bounded measurement, movement and calculation errors.

3.3 Following, flocking and capture

In these problems there are two kinds of robots in the environment: the *leader* L, and the *followers*. The leader acts independently from the others, and we can assume that it is driven by an human pilot. The followers are required to follow the leader wherever it goes (*following*), while keeping a formation they are given in input (*flocking*). In this context, a formation is simply a pattern described as a set of points in the plane, and all the robots have the same formation in input (see Figure 3).

In [26], an algorithm solving this problem has been tested by using computer simulation; the algorithm assumes no agreement. All the experiments demonstrated that the algorithm is well behaved, and in all cases the followers were able to assume the desired formation and to maintain it while following the leader along its route. Moreover, the obliviousness of the algorithm contributes to this result, since the followers do not base their computation on past leader's positions.

Finally, if the leader is considered an "enemy" or "intruder", and the pattern surrounds it, the problem is known as *capture*. Also in this case, a procedure that assumes no agreement and solves the problem has been presented in [27]. The proposed algorithm exhibits remarkable robustness, and numeric simulations indicate that the intruder is efficiently captured in a relatively short time and kept surrounded after that, as desired. Furthermore, the solution is self-stabilizing [17, 18]. In particular, any external intervention (e.g., if one or more of the cops are stopped, slowed down, knocked out, or simply faulty) does not prevent the completion of the task.

- In the *general setting* there is a procedure for the *flocking* problem [26].
- In the *general setting* there is a procedure for the *intruder* problem [27].

4 Conclusion and Discussion

In this paper, we surveyed a number of recent results on the interplay between robots' capabilities and solvability of problems. The goal of these studies is

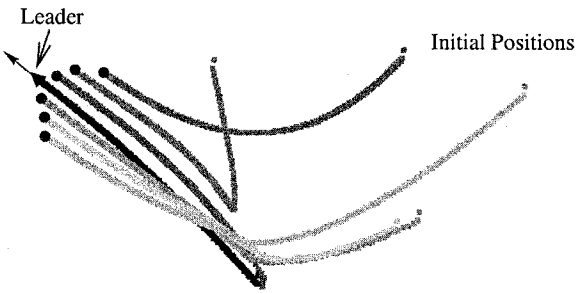

Fig. 3. Trace of the vehicles while forming and keeping a wedge shaped formation.

to gain a better understanding of the power of distributed control from an algorithmic point of view.

The area offers many open problems. The operating capabilities of our robots are quite limited. It would be interesting to look at models where the robots have more complex capabilities, e.g.: the robots have some kind of direct communication capabilities; the robots are distinct and externally identifiable; etc. Little is known about the solvability of other problems like *spreading* and *exploration* (used to build maps of unknown terrains), about the physical aspects of the models (giving physical dimension to the robots, bumping, energy saving issues, etc.), and about the relationships between geometric problems and classical distributed computations.

In the area of reliability and fault-tolerance, lightly faulty snapshots, a limited range of visibility, obstacles that limit the visibility and that moving robots must avoid or push aside, as well as robots that appear and disappear from the scene clearly are all topics that have not yet been studied.

We believe that investigations in these areas will provide useful insights on the ability of weak robots to solve complex tasks.

Acknowledgements The Authors would like to thank Paola Flocchini and Peter Widmayer for their help and suggestions in the preparation of this survey. This research is supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

1. N. Agmon and D. Peleg. Fault-tolerant Gathering Algorithms for Autonomous Mobile Robots. In *Proc. of the 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 1070 – 1078, 2004.
2. H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. *IEEE Trans. on Robotics and Automation*, 15(5):818–828, 1999.

3. T. Balch and R. C. Arkin. Behavior-based Formation Control for Multi-robot Teams. *IEEE Trans. on Robotics and Automation*, 14(6), December 1998.
4. R. Beckers, O. E. Holland, and J. L. Deneubourg. From Local Actions To Global Tasks: Stigmergy And Collective Robotics. In *Art. Life IV, 4^{th} Int. Worksh. on the Synth. and Simul. of Living Sys.*, 1994.
5. G. Beni and S. Hackwood. Coherent Swarm Motion Under Distributed Control. In *Proc. DARS'92*, pages 39–52, 1992.
6. Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng. Cooperative Mobile Robotics: Antecedents and Directions. In *Int. Conf. on Intel. Robots and Sys.*, pages 226–234, 1995.
7. I. Chatzigiannakis, M. Markou, and S. Nikoletseas. Distributed Circle Formation for Anonymous Oblivious Robots. In *Experimental and Efficient Algorithms: Third International Workshop (WEA 2004)*, volume LNCS 3059, pages 159 –174, 2004.
8. Q. Chen and J. Y. S. Luh. Coordination and Control of a Group of Small Mobile Robots. In *Proc. IEEE Int. Conf. on Rob. and Aut.*, pages 2315–2320, 1994.
9. M. Cieliebak. Gathering Non-Oblivious Mobile Robots. In *Proc. 6th Latin American Symposium on Theoretical Informatics*, pages 577–588, 2004.
10. M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the Robots Gathering Problem. In *Proc. 30th International Colloquium on Automata, Languages and Programming*, pages 1181–1196, 2003.
11. M. Cieliebak and G. Prencipe. Gathering Autonomous Mobile Robots. In *Proc. 9th Int. Colloquium on Structural Information and Communication Complexity*, June 2002.
12. R. Cohen and D. Peleg. Convergence Properties of the Gravitational Algorithm in Asynchronous Robot Systems. In *Proc. of the 12th European Symposium on Algorithms*, pages 228–239, 2004.
13. R. Cohen and D. Peleg. Robot Convergence via Center-of-Gravity Algorithms. In *Proc. of the 11th Int. Colloquium on Structural Information and Communication Complexity*, pages 79–88, 2004.
14. R. Cohen and D. Peleg. Convergence of Autonomous Mobile Robots with Inaccurate Sensors and Movements. In *Proc. 23^{th} Annual Symposium on Theoretical Aspects of Computer Science (STACS '06)*, pages 549–560, 2006.
15. R. Cohen and D. Peleg. Local Algorithms for Autonomous Robots Systems. In *Proc. of the 13th Colloquium on Structural Information and Communication Complexity*, 2006. to appear.
16. X. Défago and A. Konagaya. Circle Formation for Oblivious Anonymous Mobile Robots with No Common Sense of Orientation. In *Workshop on Principles of Mobile Computing*, pages 97–104, 2002.
17. E. W. Dijkstra. Self-stabilizing Systems in Spite of Distributed Control. *Comm. of the ACM*, 17(11):643–644, 1974.
18. S. Dolev. *Self-stabilization.* The MIT Press, 2000.
19. B. R. Donald, J. Jennings, and D. Rus. Information Invariants for Distributed Manipulation. *The Int. Journal of Robotics Research*, 16(5), October 1997.
20. E. H. Durfee. Blissful Ignorance: Knowing Just Enough to Coordinate Well. In *ICMAS*, pages 406–413, 1995.
21. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots. In *Proc. 10th International Symposium on Algorithm and Computation*, pages 93–102, 1999.

22. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of Asynchronous Mobile Robots with Limited Visibility. In *Proceedings 18th International Symposium on Theoretical Aspects of Computer Science*, volume LNCS 2010, pages 247–258, 2001.
23. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Pattern Formation by Autonomous Robots Without Chirality. In *Proc. 8th Int. Colloquium on Structural Information and Communication Complexity*, pages 147–162, June 2001.
24. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of Asynchronous Robots with Limited Visibility. *Theoretical Computer Science*, 337:147–168, 2005.
25. T. Fukuda, Y. Kawauchi, and H. Asama M. Buss. Structure Decision Method for Self Organizing Robots Based on Cell Structures-CEBOT. In *Proc. IEEE Int. Conf. on Robotics and Autom.*, volume 2, pages 695–700, 1989.
26. V. Gervasi and G. Prencipe. Coordination Without Communication: The Case of The Flocking Problem. *Discrete Applied Mathematics*, 143:203–223, 2003.
27. V. Gervasi and G. Prencipe. Robotic cops: The intruder problem. In *Proc. IEEE Conference on Systems, Man and Cybernetics*, pages 2284–2289, 2003.
28. D. Jung, G. Cheng, and A. Zelinsky. Experiments in Realising Cooperation between Autonomous Mobile Robots. In *ISER*, 1997.
29. Y. Kawauchi and M. Inaba and. T. Fukuda. A Principle of Decision Making of Cellular Robotic System (CEBOT). In *Proc. IEEE Conf. on Robotics and Autom.*, pages 833–838, 1993.
30. M. J Matarić. *Interaction and Intelligent Behavior.* PhD thesis, MIT, May 1994.
31. S. Murata, H. Kurokawa, and S. Kokaji. Self-assembling Machine. In *Proc. IEEE Conf. on Robotics and Autom.*, pages 441–448, 1994.
32. F. R. Noreils. Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment. *The Int. J. of Robot. Res.*, pages 79–98, 1993.
33. Y. Oasa, I. Suzuki, and M. Yamashita. A Robust Distributed Convergence Algorithm for Autonomous Mobile Robots. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, pages 287–292, October 1997.
34. L. E. Parker. On the Design of Behavior-Based Multi-Robot Teams. *Journal of Advanced Robotics*, 10(6), 1996.
35. G. Prencipe. On The Feasibility of Gathering by Autonomous Mobile Robots. In *Proc. 12th Int. Colloquium on Structural Information and Communication Complexity*, pages 246–261, 2005.
36. G. Prencipe. The Effect of Synchronicity on the Behavior of Autonomous Mobile Robots. *Theory of Computing Systems*, 38:539–558, 2005.
37. S. Samia, X. Défago, and T. Katayama. Convergence Of a Uniform Circle Formation Algorithm for Distributed Autonomous Mobile Robots. In *In Journés Scientifiques Francophones (JSF), Tokio, Japan*, 2004.
38. K. Sugihara and I. Suzuki. Distributed Algorithms for Formation of Geometric Patterns with Many Mobile Robots. *Journal of Robotics Systems*, 13:127–139, 1996.
39. I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *Siam J. Computing*, 28(4):1347–1363, 1999.
40. O. Tanaka. Forming a Circle by Distributed Anonymous Mobile Robots. Technical report, Department of Electrical Engineering, Hiroshima University, Hiroshima, Japan, 1992.

41. P. K. C. Wang. Navigation Strategies for Multiple Autonomous Mobile Robots Moving in Formation. *Journal of Robotic Systems*, 8(2):177–195, 1991.