On the Intelligent Behavior of Stupid Robots

G. Prencipe and V. Gervasi
Dipartimento di Informatica – Università di Pisa
Corso Italia, 40 I-56125 Pisa, Italy

Abstract. It is well known that sophisticated behavior can be exhibited by systems (or *communities*) composed of simple elements (*members*), each of which has only very limited intelligence and exhibits only simple behavior. Exploiting this *emergent behavior* in robotic systems is particularly important, since systems built according to this principle tend to cost less and be more robust and efficient than systems composed by more complex, powerful, intelligent – but less robust – units.

This paper presents a survey of some computational model for such simple robots, and discusses which problems can be successfully tackled at various levels of "stupidity" of the units.

1 Introduction

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, in the last decade 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. In a system consisting of a set of totally distributed agents the goal is generally to exploit the multiplicity of the elements in the system so that the execution of a certain predetermined task occurs in a coordinated and distributed way.

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 [7, 17].

In this paper, we concentrate on the problem of coordinating a set of mobile robots (units) that can freely move on a plane so that they collectively accomplish some given task. Generally, this problem has been approached mostly from an empirical point of view. Among the studies conducted in the engineering area, we can cite the Cellular Robotic System (CEBOT) of Kawaguchi *et al.* [14], the Swarm Intelligence of Beni *et al.* [3], the Self-Assembly Machine ("fructum") of Murata *et al.* [16], etc. A number of remarkable studies has been done also in the AI community, e.g., on social interaction leading to group behavior by Matarić [15], on selfish behavior of cooperative robots in animal societies by Parker [17], on primitive animal behavior in pattern formation by Balch and Arkin [2], to cite just a few (see [4] for a survey). In all these investigations, however, algorithmic aspects were somehow implicitly an issue, but clearly not a major concern. An investigation with an algorithmic flavor has been undertaken within the AI community by Durfee [8], 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. Also of interest in this perspective is how to measure the complexity of different solutions, and thus to gain an understanding of how efficiently the various tasks can be performed.

In many 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 first work proposed under this light is that of I. Suzuki *et. al.* [1, 20, 21] (and, with this focus, a rarity in the mobile robots literature). It approaches the algorithmic issues related to the pattern formation for robots, under several assumptions on the power of the individual robot. The only other study, to our knowledge, that approaches the problem of coordinating and controlling a set of autonomous mobile robots from a computational point of view is that of Prencipe *et al.* [9, 10, 11, 18, 19]. However, the two approaches differ with respect to the assumptions on the robots capabilities (the robots in this model are "weaker" than those in the previous one).

The paper is organized as follows. The two models cited above are presented in Section 2, and their differences are highlighted. Then, Section 3 surveys the main results obtained on the solvability of a number of behavioral problems, with a particular emphasis on the degree of knowledge that is needed by the robots to solve a problem. A brief discussion of open issues in the field, together with some conclusion, complete the paper.

2 Computational models

In this section we present two approaches to model the control and coordination of a set of autonomous mobile robots. The first one, the SYm model, was originally proposed in [21]; the second one, our CORDA model, was introduced in [9]. As already stated, these are the only two studies, to our knowledge, that tackle the problem of controlling an environment populated by a set of totally autonomous mobile robots from a computational viewpoint.

2.1 Common Features

The two models discussed in this paper share some basic features. The robots are modeled as units with computational capabilities, which are able to freely move in the plane. They are viewed as points, and they are equipped with sensors that let them observe the positions of the other robots in the plane. Depending on whether they can observe all the plane or just a portion of it, two different models can arise: *Unlimited* and *Limited Visibility* model (each robot can see only whatever is at most at distance V from it). The robots are *anonymous*, meaning that they are a priori indistinguishable by their appearances, and they do not have any kind of identifiers that can be used during the computation. They are *asynchronous* and no central control is allowed.

Each robot has its own *local view* of the world. This view includes a local Cartesian coordinate system with origin, unit of length, and the *directions* of two coordinate axes, identified as

x axis and y axis, together with their *orientations*, identified as the positive and negative sides of the axes. The robots do not necessarily share the same $x - y$ coordinate system, and do not necessarily agree on the location of the origin, or on the unit distance. They execute, however, the same deterministic algorithm, which takes in input the positions of the robots in the plane observed at a time instant t, and returns a destination point towards which the executing robot moves. The algorithm is *oblivious* if the new position is determined only from the positions of the others at t, and not on the positions observed in the past; otherwise, it is called *non oblivious*. Moreover, there are no explicit means of communication: the only means 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). For oblivious robots, even this sending of information is impossible, since the others will not remember previous positions.

Clearly, these basic features render the modeled robots simple and rather "weak", especially considering the current engineering technology. But, as already noted, the main interest in the studies done in [9, 21], is to approach the problem of coordinating and controlling a set of mobile units from a *computational* point of view. The robots are modeled as "weak robots" because in this way it is possible to formally analyze the strengths and weaknesses of the distributed control. Furthermore, this simplicity can also lead to some advantages. For example, avoiding the ability to remember what has been computed in the past gives the system the nice property of self-stabilization [10, 21].

During its life, each robot repeatedly executes a cycle composed of four phases:

1. **Look** The robot observes the world by activating its sensors which will return a snapshot of the positions of all other robots with respect to its local coordinate system. Each robot r_i is viewed as a point, and therefore its position in the plane at time t, denoted by $p_i(t)$, is given by its coordinates. Therefore, the result of a look by robot r_i at time t is a set of coordinates $S_i(t)$. In addition, the 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*[1].
2. **Compute** The robot performs a *local computation* according to its deterministic algorithm $\mathcal{A}$. For oblivious computations, $\mathcal{A}$ takes as input only $S_i(t)$ captured in the previous phase; otherwise, in the non-oblivious case, it takes as input the entire history of all the positions observed since the beginning of the computation.[2] The result of $\mathcal{A}$, denoted by $d_i(t)$, can be a destination point or a *null movement* (i.e., the robot decides to not move).
3. **Move** If the result of the computation was a *null movement*, the robot does not move; otherwise it moves towards the point $d_i(t)$ computed in the previous phase. The two models have different assumptions about the amount of movement that is performed in a single *Move*, that will be better illustrated in the following. Since the robots are viewed as points, it is assumed that two robots can occupy the same position simultaneously and never collide.
4. **Wait** The robot is idle. A robot cannot stay infinitely idle; at a certain point, the robot will go back to the first phase.

2.2 The SYm model

In this section we better describe how the movement of the robots is modeled in SYm [1, 21]. The authors assume discrete time $0, 1, 2, \ldots$. At each time instant t, every robot r_i is either

[1]Of course, in the case in which robots can detect multiplicity, $S_i(t)$ becomes a multiset of positions.

[2]Note that the non-obliviousness feature does not imply the possibility for a robot to find out which robot corresponds to which position it stored, since the robots are anonymous.

active or *inactive*. At least one robot is active at every time instant, and every robot becomes active at infinitely many unpredictable time instants. A special case is when every robot is active at every time instant; in this case the robots are *synchronized*.

For any $t \geq 0$, if r_i is inactive, then $p_i(t+1) = d_i(t) = p_i(t)$; otherwise $p_i(t+1) = d_i(t)$. The maximum distance that r_i can move in one step is bounded by a distance $\epsilon_i > 0$ (this implies that every robot is then capable of traveling at least a distance $\epsilon = \min\{\epsilon_1, \ldots, \epsilon_n\} > 0$). The reason for such a constant is to simulate a continuous monitoring of the world by the robots. However, this limit is stated as a restriction on $\mathcal{A}$, that must be written in such a way as to never return a destination point at a distance greater than ϵ_i from $p_i(t)$.

In SYm, the robots execute phases 1-4 *atomically,* in the sense that a robot that is active and observes at t, has already reached its destination point at $t+1$. Therefore, a robot takes a certain amount of time to move (the time elapsed between t and $t+1$), but no fellow robot can see it *while* it is moving (or, alternatively, we can say that the movement is *instantaneous*).

2.3 The CORDA *model*

In CORDA, time is modeled in a continuous way; the environment is *fully asynchronous,* in the sense that there is no common notion of time, and a robot observes the environment at unpredictable time instants. Moreover, no assumptions on the cycle time of each robot, and on the time each robot elapses 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. Thus, each robot can take its own time to compute, or to move towards some point in the plane: in this way, it is possible to model different computational and motorial speeds of the units. Moreover, every robot can be seen *while* it is moving by other robots that are observing.

This feature renders more difficult the design of an algorithm to control and coordinate the robots. For example, when a robot starts a *Move* state, it is possible that the movement it will perform will not be "coherent" with what it observed, since, during the *Compute* state, other robots can have moved. These properties are more formally described by the following assumptions: **A1 (Computational Cycle)** The amount of time required by a robot r_i to complete a computational cycle is not infinite, nor infinitesimally small. **A2 (Distance)** The distance traveled by a robot r_i in a *Move* is not infinite. Furthermore, it is not infinitesimally small: there exists an arbitrarily small constant $\delta_{r_i} > 0$, such that if the result of the computation is not a *null movement* and the destination point is closer than δ_{r_i}, r_i will reach it; otherwise, r_i will move towards it of at least δ_{r_i}. Therefore, in CORDA there is no assumption on the maximum distance a robot can travel before observing again (apart from the bound given from the destination point that has to be reached).

2.4 Discussion: main differences between SYm and CORDA

The main differences between the two models are, as stated before, in the way the asynchronicity is regarded and in the movement model.

In SYm, a robot r_i that observes a fellow robot r_j at time t in a position $p_j(t)$ can confidently assume that r_j will not move before having observed the environment at a time $t_1 > t$. In other words, the future evolution of the group of robots is determined exclusively by the current (i.e., observed at time t) positions of all the units. This means that r_i, based exclusively on knowledge of its own algorithm (that is the same for all the robots) and on what is observed at time t, has *complete information* about the future behavior of all the units.

On the other hand, in CORDA a robot r_i can observe r_j at time t while it is moving. r_i cannot know what the destination of r_j will be, since that destination was determined by an observation made by r_j at some time $t_0 < t$ — an information that is irremediably lost to r_i. Thus, r_i has only *incomplete information* about the behavior of its fellows.

Regarding movement, in SYm an active robot always travels at most a distance ϵ_i in each cycle, since $\mathcal{A}$ must never try to move farthest than that. In CORDA, instead, there is only a lower bound on such distance, and no restriction is placed on $\mathcal{A}$: when a robot r_i moves, it moves at least some positive, small constant δ_{r_i}. The reason for this constant is to better model reality: it is not realistic to allow the robots to move an infinitesimally small distance.

3 Solvable problems and limitations

Despite the simplicity of the models described in the previous section, there is a surprisingly large class of problems that can be stated in our framework. The most studied group concerns *geometric* problems, like forming a certain pattern, following a trail, or deploy the robots in order to guarantee optimal coverage of a certain terrain.

Also, many classical problems from the distributed algorithms field can be reformulated as geometric problems in our model. For example, election of a leader among a set of homogeneous agents can be simulated by asking the robots to form a certain pattern, and designating the robot occupying a special position as the leader.

In the following, we survey the results obtained so far in solving a number of problems according to this new approach. It should be noted that, although the results have a relevance on their own from a computational point of view, they are also relevant in robotics, and in some cases in understanding emerging behavior in other (e.g., animal) communities.

3.1 Arbitrary 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 [5, 20, 21, 22]). The basic research questions are which patterns can be formed, and how they can be formed. These questions have been studied mostly from an empirical point of view, with no actual proofs of correctness. Actually, many solutions do not terminate and they never form the desired pattern (the robots just converge towards it); such solutions are called "*convergence*".

Unlike previous work, we are interested in (provably correct) "formation" solutions: solutions which *always* form the pattern within finite time; we have been studying what patterns can be formed and how, within this context.

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: for example, heavy loads can be moved by a group of robots that form a pattern corresponding to the shape of the load.

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.

In [21], Suzuki and Yamashita examine the problem in SYm, characterizing what kind of patterns can be formed, but their results and all their algorithms hold considering non-oblivious and synchronous robots; in fact, they require an unbounded amount of memory at each robot to remember all of the past. Their result characterizes the patterns in terms of the central symmetry structures in the initial positions of the robots and of the corresponding symmetry in the input pattern. In detail, if there is no central symmetry in the initial configuration, the robots can form any arbitrary pattern. However, an unfortunate error in their Lemma 4.2 affects both the correctness of the algorithm they propose, and the classification of the patterns that can be formed in the case where central symmetry is present.

On the contrary, in CORDA the robots are completely oblivious and asynchronous. The following theorem summarizes the results holding for a set of n robots in CORDA that at the beginning are placed on distinct positions:

Theorem 3.1.

1. *With a common coordinate system of (i.e., the robots agree on) both x and y directions and orientations, the robots can form an arbitrary given pattern [9]. This is equivalent to assuming that the robots are equipped with a compass.*
2. *With agreement on only one axis direction and orientation, the pattern formation problem is unsolvable when n is even, while it can be solved if n is odd [9].*
3. *With agreement on only one axis direction and orientation, an even number of robots can form only symmetric patterns that have at least one axis of symmetry not passing through any vertex of the pattern [12].*
4. *With no agreement at all, the robots cannot form an arbitrary given pattern [9].*

3.1.1 Homing and gathering

In the gathering problem, the robots, initially placed in arbitrary positions, are required to gather in a single point. If the point is fixed in advance this behavior is called "homing", otherwise we refer to the problem as "gathering". The homing problem is of course easier, since each robots knows beforehand where to go. Most studies have thus concentrated on the gathering problem.

In the unlimited visibility setting of both CORDA and SYm, one feature the robots must have in order to solve this problem, is the ability to detect *multiplicity*, that is the ability to detect if on a given point on the plane there is more than one robot [19] (recall that the robots are viewed as points).

In SYm, the solvability of the gathering problem is given by the following

Theorem 3.2.

1. *if the robots have unlimited visibility, the problem is unsolvable for $n = 2$ [21];*
2. *if the robots have unlimited visibility, there exists an oblivious algorithm for solving the gathering problem for a number of robots $n \geq 3$ [21];*
3. *it the robots have limited visibility, there exists an oblivious procedure that lets robots converge towards (but not necessarily reach) a point for any n [1].*

In CORDA, on the other hand, all the results obtained so far concern proper convergence — in other words, when the problem is solvable there is a guarantee that the robots will reach the gathering point in a finite number of moves:

Theorem 3.3.

1. *if the robots have unlimited visibility, the problem is unsolvable for $n = 2$ [6];*
2. *if the robots have unlimited visibility, there exists an oblivious algorithm for solving the gathering problem for $n = 3$ or $n = 4$ [6];*

3. *if the robots have unlimited visibility, there exists an oblivious algorithm for solving the gathering problem for $n \geq 5$ if the initial configuration of the robots is not* biangular[3] *[6];*
4. *if the robots have limited visibility, there exists an oblivious algorithm for solving the gathering problem for any n if the robots agree on a common coordinate system [11].*

In the limited visibility setting of both SYm and CORDA, a 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 [1, 11].

3.2 Following, flocking and capture

In this problem, 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. Finally, if the leader is considered an “enemy” or “intruder”, and the pattern surrounds it, the problem is known as *capture*.

To our knowledge, this family of problems has only been analyzed in CORDA, assuming that the robots have no agreement on the orientation and direction of the x and y axis. Let us denote with ρ the *radius* of the pattern, that is the maximum distance of a point in the pattern from the position of the leader, with v_L and v_r respectively the maximum linear velocity of the leader and of the generic follower r, and with ω_L the maximum angular velocity of L. The following theorem states the condition under which the followers can successfully follow the leader.

Theorem 3.4 ([13]). *In order for the formation to be kept in movement, it is necessary for the leader to have $\omega_L \cdot \rho + v_L < \min_r v_r$.*

In [13], an algorithm solving this problem in CORDA has been tested by using computer simulation. 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.

4 Conclusions and open issues

In this paper, we surveyed a number of recent results on a new approach to analyze the amount of intelligence and knowledge required to simple robots in order to accomplish complex tasks. In particular, we concentrated on solvability results for a large class of geometric problems, under two different models of the behavior of the robots. These results are important in a number of practical applications, and shed a new light on the theoretical limits of the models.

The area offers many open problems. Little is known about the solvability of other geometrical 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 geometrical problems and classical distributed algorithms. We believe that further investigation in this area is needed, and will provide useful insights on the ability of stupid robots to solve hard tasks.

[3] n distinct points in a plane are in a *biangular* configuration if there exists a point c, two angles α and β, and an ordering of the points π such that, for any i, $\pi_i \widehat{c} \pi_{i_+} = \alpha$ and $\pi_{i_+} \widehat{c} \pi_{i_{++}} = \beta$, where $i_+ = (i+1) \mod n$.

References

[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. *IEEE Trans. on Rob. and Autom.*, 15(5):818–828, 1999.

[2] T. Balch and R. C. Arkin. Behavior-based Formation Control for Multi-robot Teams. *IEEE Trans. on Rob. and Autom.*, 14(6), 1998.

[3] G. Beni and S. Hackwood. Coherent Swarm Motion Under Distributed Control. In *Proc. DARS'92*, pages 39–52, 1992.

[4] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng. Cooperative Mobile Robotics: Antecedents and Directions. In *IEEE/TSJ Int. Conf. on Intell. Robots and Sys.*, pages 226–234, 1995.

[5] Q. Chen and J. Y. S. Luh. Coordination and Control of a Group of Small Mobile Robots. In *Proc. of IEEE Int. Conf. on Rob. and Autom.*, pages 2315–2320, 1994.

[6] M. Cieliebak and G. Prencipe. Gathering Autonomous Mobile Robots. In *Proc. SIROCCO*, June 2002.

[7] B. R. Donald, J. Jennings, and D. Rus. Analyzing Teams of Cooperating Mobile Robots. Technical Report TR 94-1429, Dept. of Computer Science, Cornell University.

[8] E. H. Durfee. Blissful Ignorance: Knowing Just Enough to Coordinate Well. In *ICMAS*, pages 406–413, 1995.

[9] 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. ISAAC*, volume LNCS 1741, pages 93–102, December 1999.

[10] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Distributed Coordination of a Set of Autonomous Mobile Robots. In *IEEE Intelligent Veichle Symposium (IV 2000)*, pages 480–485, 2000.

[11] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of Autonomous Mobile Robots With Limited Visibility. In *Proc. STACS*, volume LNCS 2010, pages 247–258, February 2001.

[12] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Pattern Formation by Autonomous Robots Without Chirality. In *Proc. SIROCCO*, pages 147–162, June 2001.

[13] V. Gervasi and G. Prencipe. Need a Fleet? Use The Force! In *FUN With Algorithms 2 (FUN 2001)*, pages 149–164, May 2001.

[14] Y. Kawauchi and M. I. and. T. Fukuda. A Principle of Decision Making of Cellular Robotic System (CE-BOT). In *Proc. IEEE Conf. on Rob. and Autom.*, pages 833–838, 1993.

[15] M. J. Matarić. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. In *From Animals to Animats 2: Int. Conf. on Simulation of Adaptive Behavior*, pages 423–441. The MIT Press, 1993.

[16] S. Murata, H. Kurokawa, and S. Kokaji. Self-Assembling Machine. In *Proc. IEEE Conf. on Rob. and Autom.*, pages 441–448, 1994.

[17] L. E. Parker. Adaptive Action Selection for Cooperative Agent Teams. In *Proc.* 2^{nd} *Int'l. Conf. on Simulation of Adaptive Behavior*, pages 442–450, 1992.

[18] G. Prencipe. CORDA: Distributed Coordination of a Set of Autonomous Mobile Robots. In *Proc. ERSADS*, pages 185–190, May 2001.

[19] G. Prencipe. *Distributed Coordination of a Set of Autonomous Mobile Robots*. PhD thesis, Università di Pisa, 2002. http://sbrinz.di.unipi.it/~peppe/tesi.ps.

[20] K. Sugihara and I. Suzuki. Distributed Motion Coordination of Multiple Mobile Robots. In *Proc. IEEE Int'l Symp. on Intelligent Control*, pages 138–143, 1990.

[21] I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *Siam J. Comput.*, 28(4):1347–1363, 1999.

[22] P. K. C. Wang. Navigation Strategies for Multiple Autonomous Mobile Robots Moving in Formation. *J. of Rob. Sys.*, 8(2):177–195, 1991.