Spatial awareness in robotic swarms through local wireless communications

(Extended Abstract)

Frederick Ducatelle
Dalle Molle Institute for Artificial Intelligence (IDSIA)
Lugano, Switzerland
frederick@idsia.ch

Gianni A. Di Caro
Dalle Molle Institute for Artificial Intelligence (IDSIA)
Lugano, Switzerland
gianni@idsia.ch

Luca M. Gambardella
Dalle Molle Institute for Artificial Intelligence (IDSIA)
Lugano, Switzerland
luca@idsia.ch

ABSTRACT

We propose a fully distributed approach to endow robots in a swarm with awareness of their relative position with respect to the rest of the swarm. Such spatial awareness can be used to support spatially differentiated task allocation or for pattern formation. The approach we propose only relies on local communications and is based on a combination of distributed consensus and load balancing. We test the effectiveness of our algorithm in extensive simulation tests and we also validate it in experiments with real robots.

Categories and Subject Descriptors

I.2.9 [**Robotics**]; I.2 [**Distributed Artificial Intelligence**]: Coherence and coordination; C.2 [**Computer Communication Networks**]: Distributed applications

General Terms

Algorithms

Keywords

Swarm robotics, geometric bisectioning, spatial aggregation

1. INTRODUCTION

The aim of this work is to endow robots in a swarm with awareness of their relative position with respect to the rest of the swarm. Such *spatial awareness* can be used to support spatially differentiated task allocation (e.g., split the swarm in different, spatially close, groups, and let each group engage in a different task, such as exploring different regions of an environment), or for pattern formation, among others. The task we focus on is to assign the robots of the swarm to two different classes, C_0 and C_1, in such a way that *the two classes are spatially segregated*: the robots in class C_0 are found on one side of the swarm, and the robots in class C_1 on the other side of the swarm. The problem that we are solving can be formally described as follows. Let $G(V, E)$ be a Euclidean graph where the node set V represents geometric entities, such as robots, positioned in the plan. Nodes are able to communicate with each other over a wireless medium. Two nodes i and j are connected by a link $(i, j) \in E$ if: (i) their Euclidean distance is less than or equal to the maximum communication range R_{max} (*range-constrained connectivity*), and (ii) no major occlusions are present between the two nodes (*line-of-sight communication constraint*). Each node only knows about its neighbors, no other network information is assumed. The objective is to find, adopting a fully decentralized approach, a geometric partitioning of the graph in k classes, where each class contains (approximately or precisely) the same number n_k of nodes, and the nodes in each partition are geometrically close to each other. We focus on the case $k = 2$. Figure 1 illustrates different partitionings in two classes. We aim to obtain partitionings like in (a) and (b).

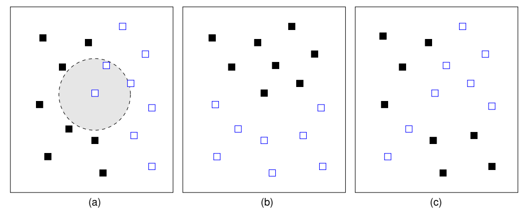

Figure 1: Examples of different ways to realize a geometric partitioning in two classes (indicated by the black and white squares) given a communication range (indicated by the grey disk in (a)).

To solve this problem, we look for an algorithm that is robust, scalable, efficient, works in a decentralized way, and has limited requirements in terms of available sensor or actuators. We rule out the use of global positioning information (not always available, especially in indoor environments) as well as the use of physical mobility (not always possible, slow, and energy-greedy). Instead, we propose an algorithm which uses only *local communication.* The robots/nodes only need to be able to identify their neighbors and communicate with them. Only a relatively *low channel bandwidth* is required to let the algorithm working effectively. We consider the general case of robots/nodes equipped with a wireless communication interface. The algorithm combines elements from different approaches to similar problems: algorithms for solving *minimum bisection problems* [1]; algorithms for *swarm robotics aggregation* [3], and distributed algorithms for *consensus load balancing* [2].

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. ALGORITHM OVERVIEW

The degree of membership of a robot i to one of the two classes C_0 and C_1 is represented by using *load variables* $u_i \in [0, 1]$. $u_i = 1$ means full membership of i to class C_1, $u_i = 0$ means membership to class C_0, while intermediate values indicate different degrees of membership to C_0 and C_1. At the start, each robot i decides with a probability of 0.5 whether it is loaded or not, and sets accordingly a variable u_i. Each robot i also keeps a value $v_i \in [0, 1]$, which is an estimate of *how loaded on average* the robots in its neighborhood are. After the initialization, the robots start to communicate locally, with two goals: to update the estimate v_i, and to let loads travel through the swarm, until they stop at different robots. Local values of v_j variables decide when to leave, where to go, and where to stop. Load traveling across the robot network goes in a number of phases. Each phase aims, in different ways, to eventually create a single connected cluster of robots of class C_1 which is spatially well separated from the cluster of unloaded robots (i.e., of class C_0), as illustrated in Figure 1 (a) and (b).

In **phase 0**, following the first creation or the reception of a load, the load leaves robot i if i's neighborhood stays unloaded for a certain amount of time (i.e, v_i is less than a threshold v_{min}). **Phase 1** is a *steepest ascent* with respect to current load distribution: the load is iteratively sent to the neighbor j with the highest v_j, until the local maximum is reached. **Phase 2** is a *steepest descent*: the load moves to a local minimum of v_j, meaning that it looks for an area which is unloaded. **Phase 3** is again a *steepest ascent*: the load greedily looks for a new loaded area, possibly with a higher value of v_j than that of phase 1. **Phase 4** is a *slowest descent*: the load moves from robot to robot to decreasing values of u_i, until it reaches an unloaded robot ($u_j = 0$), where it moves back to phase 0. The idea is that the slow descent will make the loads rather go towards areas where there are only a few unloaded nodes, so that the load goes to fill small empty pockets. If no unloaded robot is found before reaching a local minimum, the load takes a random step, and returns to phase 1: start all over again. Once the load has reached phase 0 at a robot i, it sets the local value u_i to 1. If more loads cluster around robot i, the local value v_i will grow, keeping the load stationary at robot i. In this way, $v_i \geq v_{min}$, preventing the load to leave i and letting the cluster grow further. Eventually, loads stop moving, converging in two spatially separated clusters.

3. EXPERIMENTAL RESULTS

We ran simulation tests considering as reference robot the *foot-bot*, developed during the *Swarmanoid* project (`http://www.swarmanoid.org`). For the work presented here, the relevant on-board device is the infrared-based *range-and-bearing* that provides *line-of-sight* communication. It sends messages of 10 bytes at a rate of 10 messages per sec.

Each simulation test (50 trials per test) runs for 1200 time steps = 2 minutes. We measure two things: *linear separability* and *imbalance*. Linear separability is evaluated by fitting a line to the space in which the robots are placed, in such a way that the loaded robots are found on one side of the line and the unloaded ones on the other side. Results for linear separability range from 0 (optimal) to 0.5 (worst). The imbalance evaluates whether the two classes are of the same size. We report the number of robots in the smallest of the two classes, divided by the total number of robots in the swarm. The optimal value is 0.5, the worst possible is 0.

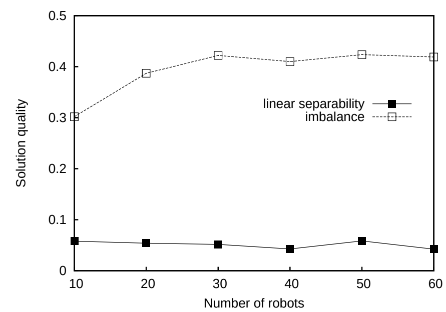

Figure 2: Experiments with varying number of robots maintaining constant the area.

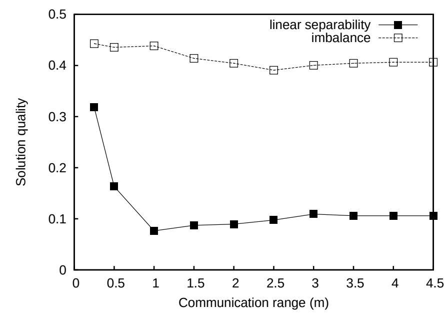

Figure 3: Experiments with varying communication range maintaining fixed to 50 the number of robots.

In a first series of tests, we vary the *number of robots* in the swarm, from 10 up to 60. The communication range of the robots is limited to 1 m. The results in Figure 2 show that the algorithm works quite well in separability, and is robust with respect to the number of robots, although for the smallest swarms, results become a bit less good because of less connectivity and too few loads around.

In a second series of tests, we vary the *communication range*, from 0.25 up to 4.5 m, fixing the the number of robots to 50. The results in Figure 3 show that the algorithm works badly at short communication ranges, due to the fact that the communication network gets disconnected. For medium and high communication ranges, the results are very good.

We also ran a limited set of experiments using a small swarm of 15 foot-bots deployed in different initial configurations. Sample videos are available here: `http://www.idsia.ch/~gianni/SwarmRobotics/GeometricSplitting.html`.

4. REFERENCES

[1] R. Battiti and A. Bertossi. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Transactions on Computers*, 48(4), April 1999.

[2] M. Franceschelli, A. Giua, and C. Seatzu. A gossip-based algorithm for discrete consensus over heterogeneous networks. *IEEE Transactions on Automatic Control*, 55(5):1244–1249, 2010.

[3] S. Garnier, C. Jost, R. Jeanson, J. Gautrais, M. Asadpour, G. Caprari, and G. Theraulaz. Collective decision-making by a group of cockroach-like robots. In *Proceedings of IEEE Swarm Intelligence Symposium (SIS)*, pages 233–240, 2005.