Journal of Computational Science 31 (2019) 86–94

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

# PSO-based strategy for the segregation of heterogeneous robotic swarms☆

Fabrício R. Inácio, Douglas G. Macharet, Luiz Chaimowicz

*Computer Vision and Robotics Laboratory (VeRLab), Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, MG, Brazil*

ARTICLE INFO

*Article history:*
Received 1 August 2018
Received in revised form
23 November 2018
Accepted 14 December 2018
Available online 29 December 2018

*Keywords:*
Swarm robotics
Group segregation
ORCA
PSO

ABSTRACT

A robotic swarm is a particular type of multi-robot system that employs a large number of simpler agents in order to cooperatively perform different types of tasks. A topic that has received much attention in recent years is the idea of segregation. This concept is important, for example, in tasks that require maintaining robots with similar features or objectives arranged in cohesive groups, while robots with different characteristics remain separated in their own groups. In this paper, we propose a decentralized methodology to segregate heterogeneous groups of robots that are randomly distributed on the environment. Our approach to segregate robots consists of extending the Optimal Reciprocal Collision Avoidance (ORCA) algorithm with a navigation strategy inspired on the Particle Swarm Optimization (PSO). A series of simulations in different scenarios shows that the groups were able to successfully converge to a segregated state in all the evaluated cases. Furthermore, the methodology allowed for a faster convergence of the groups when compared to the state-of-the-art technique.



## 1. Introduction

The use of multi-robot systems in different contexts can bring several advantages over single-robot ones. In this sense, we consider the specific scenario of robotic swarms, which are systems composed of a large number of agents seeking to cooperatively accomplish a particular task. Inspired by colonies of social insects that cooperate with each other to carry out tasks of common interest, robotic swarms emerged as an alternative to solve complex problems.

In general, swarms consist of simpler agents with little processing power and capable of performing a limited set of actions. Another common feature in swarms is the limitation of perception and communication between agents. Normally, each agent is able of sensing a small portion of the environment and communicating locally, i.e., communication takes place only between agents that are close to each other.

Similarly to various swarms found in nature (e.g. some insects colonies, bird flocks or fish schools), an important aspect in robotic swarms is the absence of a central entity responsible for coordinating the entire swarm. Hence, each individual behavior contributes to emergent group behaviors that guide the swarm to its objectives [1].

One of these group behaviors is segregation, a natural phenomenon that is commonly used as a sorting mechanism by several biological systems and can be useful in many different tasks and scenarios. This behavior can be important for maintaining robots with similar features or objectives arranged in cohesive groups, while groups with different characteristics remain separated [2].

The objective of this work is to propose a technique that is able to separate the agents into groups considering an initial scenario with a swarm randomly distributed in the environment (as illustrated in Fig. 1).

In this paper, we propose an evolutionary-based strategy to segregate a swarm composed of different groups of robots. Our approach is based on the PSO [3] method to use the knowledge of the neighbors of each agent in order to optimize the process of choosing which velocities the agents must develop in order to group and stay with the individuals of their kind. These velocities are then passed to the ORCA [4] algorithm which ensures a collision-free navigation. Unlike other works found in the literature, the proposed technique is totally decentralized and assumes local perception only.

The remainder of this paper is organized as follows: Section 2 presents a literature review regarding navigation and collision avoidance approaches for multi-robot systems, and group segregation strategies for robotic swarms. The proposed methodology is presented in detail in Section 3 and validated by a set of experiments in a simulated environment, the results of which are shown

☆ Work developed with the support of CNPq, CAPES and FAPEMIG.
*E-mail addresses:* fabricio.rod@dcc.ufmg.br (F.R. Inácio), doug@dcc.ufmg.br (D.G. Macharet), chaimo@dcc.ufmg.br (L. Chaimowicz).

https://doi.org/10.1016/j.jocs.2018.12.008
1877-7503/

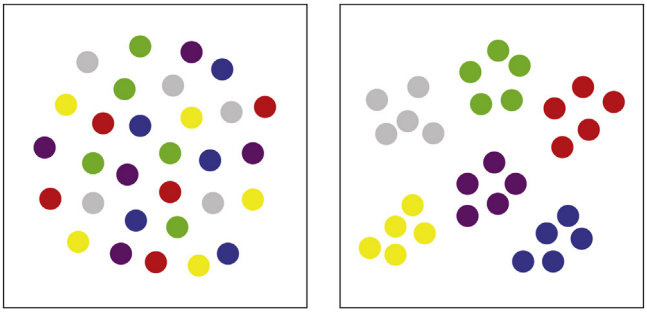



**Fig. 1.** Problem illustration. (a) Initial scenario with assorted agents randomly distributed. (b) Final condition with agents segregated into groups.

and discussed in Section 4. Finally, Section 5 concludes with future research directions.

## 2. Related work

### 2.1. Navigation and collision avoidance

The determination of paths for mobile agents is paramount in any task that requires the navigation from an initial to a goal position in the environment in a safe and efficient manner. However, most path planners consider single-agent systems, a static and known environment with a given map, and may not be able to adapt to dynamic aspects that may arise during the execution of the path.

Several reactive navigation strategies can be found in the literature to overcome such limitations, for example, Artificial Potential Fields [5], Vector Field Histogram [6] and the Dynamic Window Approach [7].

When large groups of robots move in a shared environment, efficient collision avoidance algorithms must be used to allow for a safe and effective navigation. In recent years, a set of algorithms based on reactive collision avoidance in a velocity space has been successfully used for collision avoidance in robotic swarms [4,8,9].

A seminal work in this area is the Velocity Obstacles (VO) algorithm [9]. This technique uses the agents' velocities and obstacles' positions to calculate the VO-set of all velocities that, if applied to the agents, will result in a collision and then must be avoided (they are considered obstacles in the velocity space). Next, linear and angular speeds that do not belong to the VO-set must be chosen, which ensures a safe navigation to the agents. Several extensions over the classical VO algorithm have been proposed in the literature [10–12], which aim to improve its performance and increase its possibilities of use.

One of the most successful extensions of the VO algorithm is the ORCA algorithm [4]. This technique computes for each other robot a half-plane of allowed velocities. A robot must select its optimal velocity from the intersection of all permitted half-planes, which can be done efficiently by solving a low-dimensional linear system. Moreover, differently from the VO, the ORCA algorithm guarantee local collision-free navigation.

In spite of being able to safely and efficiently navigating large groups of robots, these algorithms do not focus in moving different groups in a cohesive fashion and segregated from each other, as will be discussed in the next section.

### 2.2. Group segregation in robotic swarms

A characteristic found in social insect communities is the existence of specific roles that segregate the individuals into distinct groups (e.g. bees and ants). Each group of individuals may have a distinct set of activities and responsibilities that promote the perpetuation of the colony [13].

In robotic swarms, as well, it might also be interesting to keep different groups of agents segregated, allowing, for example, to assign specific tasks according to certain characteristics of each group.

One of the first works to deal with this problem was proposed by Kumar et al. [14]. It presented an algorithm capable of segregating distinct types of agents by applying different potential functions on a robot according to the type of agents in its neighborhood. The technique produced satisfactory results, however, it can only be applied in scenarios where there are only two different types of agents. In applications that use a very large number of robots, there may be the need to segregate the swarm in more than two groups. This technique was later extended in Santos et al. [2] in order to deal with more than two types of robots. In order to calculate the potential to be used, each agent evaluates whether the neighbors are of their same type or of a different type. Thus, it is possible to obtain the artificial force which must be applied to each agent in order to make the robots with the same type come closer, while robots of different types remain spaced apart. The main restrictions of this technique are the need for global sensing and a balanced number of agents in all groups.

More recently, an approach based on abstractions and using an artificial potential function to segregate the groups was proposed Ferreira Filho and Pimenta [15]. In spite of mathematically guaranteeing that the system will always converge to a state where multiple dissimilar groups are segregated, the approach is not completely decentralized, in the sense that robots need information from the robots of its own group and from neighboring groups most of the time. Moreover, it does not have a strategy to avoid collision among the robots. A different segregation algorithm, based on the *Brazil Nut Effect* is discussed in Groß et al. [16]. A distributed controller considers robots as having distinct virtual sizes, and local interactions make "larger" robots move outwards, segregating the robots in annular structures.

Some works have tackled the problem of segregated navigation, in which the robots start on a segregated state and have to maintain the segregation during navigation. For example, the strategy proposed in Santos and Chaimowicz [17] combines the concepts of flocking [18] and the use of abstractions to represent the groups. Built upon the classical VO algorithm [9], it introduces a novel concept called VGVO. In this algorithm, a robot $i$ senses the relative position and velocity of every robot $j$ within its neighborhood $n_i$ and builds shapes containing other teams of robots, with the exception of its own. In the workspace of robot $i$, these shapes are considered as virtual obstacles moving at the average velocity of their respective underlying robots. Thus, robot $i$ can build a virtual velocity obstacle specifying all velocities that will lead to a collision with these shapes, assuming that they maintain their current average velocities. However, the cost of computing the virtual shape for each group can be prohibitive, slowing down the navigation. A more efficient approach was proposed in Inácio et al. [19], where the ORCA algorithm was augmented with a modified version of the classical flocking behaviors to keep robots segregated while navigating.

In this paper, we propose a decentralized methodology in order to segregate heterogeneous groups of robots. As will be presented in the next section, we propose a modification over the PSO algorithm [3] and combine it with the ORCA algorithm [4] in order to avoid collisions and keep the group cohesion. Differently from others, our approach relies only in local information, obtained through sensing or communication, and is able to effectively segregate various groups of different sizes while avoiding collisions among the robots.

## 3. Methodology

In order to segregate the robots into groups of agents of the same type, we developed an approach that combines a modified version of the Particle Swarm Optimization [3] method and the ORCA algorithm [4].

Considering local sensing only, each robot is able to perceive its neighborhood and compute the velocity that will guide it to the goal (i.e., grouped with the other robots). This velocity is further used as the preferred velocity by the ORCA algorithm, which is responsible for a safe and efficient navigation.

As in both original algorithms, we assume that an agent has access to the position and velocity of all other agents present in its neighborhood, or can infer these values based on its observations. They can also detect if its neighbors belong to its own group or not. Finally, we consider that the robots start randomly distributed in the environment. The next sections detail each step of our methodology.

### 3.1. Theoretical formalization

Initially, we formalize our representation for the robots and groups. We consider a scenario in which a swarm $\boldsymbol{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_\eta\}$ of $\eta$ robots navigate in a static 2D environment. Each robot $i$ is represented by its pose $q_i = \langle x_i, y_i \rangle$, with kinematic model given by $\dot{q}_i = u_i$. The robots considered in this work are holonomic. Therefore, they can move, instantly, in any direction given by the velocity vector $u_i$.

The entire swarm is formed by distinct types (groups) of robots, which we represent by the partition $\Gamma = \{\Gamma_1, \ldots, \Gamma_m\}$, where each $\Gamma_k$ contains all agents of type $k$. We assume that $\forall j, k : j \neq k \rightarrow \Gamma_j \cap \Gamma_k = \emptyset$, i.e., each robot is uniquely assigned to a single type.

We define a robot's neighborhood based on its sensing range. During navigation, robot $i$ maintains a neighborhood $\mathcal{N}_i$ around its current position. A neighborhood consists of a circular region of radius $\lambda$ around the current position of a robot. Within the neighborhood, we may have robots from the same group and robots from other different groups. Therefore, we define the set of all robots of the same group currently located in the neighborhood of $\mathcal{R}_i$ as

$$\mathcal{N}_i^+ = \{\mathcal{R}_j : \|p_j - p_i\| \leq \lambda \wedge \mathcal{R}_j, \mathcal{R}_i \in \Gamma_k\}. \quad (1)$$

Similarly, we define the set of all robots of other groups in the neighborhood of $\mathcal{R}_i$ as

$$\mathcal{N}_i^- = \{\mathcal{R}_j : \|p_j - p_i\| \leq \lambda \wedge (\mathcal{R}_j \in \Gamma_u \wedge \mathcal{R}_i \in \Gamma_k, u \neq k)\}. \quad (2)$$

Our goal is to develop an algorithm able to segregate (cluster) robots of different types into $m$ distinct groups in the workspace, so that each group contains only agents of a single type. We consider a group segregated if agents of the same group are closer to each other than to agents of other groups [14,2]. That is, given any two different groups of agents, e.g., A and B, the mean distance between agents of the similar types (type A or type B) is less than the mean distance between agents of the different types (i.e. between type A and type B agents). More formally, we should have:

$$d_{AA} < d_{AB} \text{ and } d_{BB} < d_{AB}, \quad \forall\, \Gamma_A, \Gamma_B \in \boldsymbol{R}, \quad (3)$$

where $d_{XY}$ is the average distance between the agents of types $X$ and $Y$. Formally:

$$d_{XY} = \frac{1}{|\Gamma_X|} \sum_{i \in \Gamma_X} \left( \frac{1}{|\Gamma_Y|} \sum_{j \in \Gamma_Y} (p_i - p_j) \right). \quad (4)$$

### 3.2. Particle Swarm Optimization

The Particle Swarm Optimization (PSO) [3] is a population-based metaheuristic that iteratively tries to improve a set of candidate solutions represented by particles in the solution space. Each particle basically consists of:

- The representation of a candidate solution;
- A velocity associated to how far the particle is from a target solution.

A particle's velocity is determined considering two *best* values: (i) personal best ($p_{best}$), which is the best solution it has achieved so far; (ii) global best ($g_{best}$), the best solution obtained by any particle in the population so far.

Inspired by the classical PSO algorithm, we consider a sum of components to define the action that will be taken by an agent. In our proposed algorithm, the preferred velocity $v_k$ to be calculated by the agent is composed of three components that can be defined as follows:

- $\boldsymbol{v_{k-1}}$: The first component is the previous velocity and is associated with the inertia of the agent. This component keeps the agents moving, especially when they are not able to see any other agents of their own group in their neighborhood.
- $\boldsymbol{v_{sensing}}$: The second one is related to the information that is captured by the agent itself. The agent identifies the individual of the same type present in his neighborhood that can see the largest number of neighbors in the group. Thus, agents tend to move to the position of neighbors who have a better view of the group as a whole, which leads to the creation of clusters of agents in certain regions of the workspace.

As the agents have only local perception, being able to see a small part of the environment, usually the appearance of agent subgroups of the same type occurs in different regions of the workspace. To circumvent this problem, the third component of the algorithm was defined.

- $\boldsymbol{v_{memory}}$: This component represents the memory of each agent. During its motion, an agent stores the position in which it encountered a subgroup other than its own and the size of that subgroup. This information is updated whenever that agent encounters a subgroup larger than or equal to a subgroup previously seen. Each agent may store a single information for each group that is different from its own, i.e., if the swarm consists of five groups, each agent can store four positions, one for each group that is different from its own group. In this way, an agent may communicate with its neighbors to obtain the location in which that neighbor met a subgroup of agents of their type and move to that position in order to increase the chances of joining a subgroup. This sort of *shared memory* obtained through *local communication* considerably impacts the convergence time, as will be shown in the experiments.

We consider each agent present in the swarm as a particle. In this way, we can make a direct relation between our approach and the traditional PSO method. As mentioned earlier, the component $v_{k-1}$ helps to keep the agents moving, being directly related to the inertia of the particle defined in the PSO. The component $v_{sensing}$ operates on the information obtained by the agent through its sensors, so we can compare it with the $p_{best}$ component of the PSO. Finally, we can relate the $g_{best}$ component of the PSO method to our $v_{memory}$, since the latter acts as a desired position estimate for all agents of a similar type in a global frame.

As will be further explained in the next section, after defining the three components described above, they are summed and the result is informed to the ORCA algorithm as the preferred velocity for the agent. Upon receiving this value, the ORCA calculates the velocity closest to the preferred velocity that guarantees a collision-free navigation, and that velocity is then executed by the agent.

The PSO algorithm, and some of its variations, may have their convergence guaranteed by imposing specific ranges or values for the parameters that control their behavior [20–22]. Since our approach is a combination between the ORCA algorithm and a speed selection approach inspired by the PSO algorithm, our algorithm cannot have its convergence proven in a similar way to the works that provide such evidence. The algorithm developed uses the velocity calculated in a similar way to what happens in the PSO as a preferential speed for the ORCA. However, the velocity that is actually developed by the agents is determined by the ORCA and can be different from the one previously estimated by the based method in the PSO. However, since agents will continuously keep moving, the probability of finding a lost member of the group, or another agent that has seen it, progressively grows and aims to convergence. In this sense, considering an amount of time that tends to infinity, our method can be considered as probabilistic complete.

Furthermore, another fact that should be highlighted is the improvement given to the system by the local exchange of messages between agents. With the use of communication between the agents, we can significantly improve the performance of the system, as was verified in the experiments.

### 3.3. Setting robot velocities

The control input, i.e., the preferred velocity, is composed by the weighted sum of the three components defined in the previous section, as follows:

$$v_k = \alpha \cdot v_{k-1} + \beta \cdot v_{sensing} + \gamma \cdot v_{memory}, \tag{5}$$

where $v_{k-1}$ is the velocity of the agent in the previous iteration, $v_{sensing}$ is the velocity that attracts the agent to the position of the neighbor belonging to its group capable of seeing more neighbors from the same group and $v_{memory}$ is the velocity that leads the agent to the position reported by a neighbor belonging to a different group. Constants $\alpha$, $\beta$ and $\gamma$ are weights that may be set according to the scenario faced by the robot and prior to the start of execution.

### 3.4. Collision Avoidance – ORCA

The Optimal Reciprocal Collision Avoidance (ORCA) algorithm [4] is a navigation strategy based on the concept of velocity obstacles [9].

Consider two robots $\mathcal{R}_A$ and $\mathcal{R}_B$ with radii $r_A$ and $r_B$, positions $p_A$ and $p_B$ and velocities $v_A$ and $v_B$, respectively. Robot $\mathcal{R}_A$ tries to reach an assigned goal position $g_A$ by selecting a *preferred velocity* $v_A^{pref}$. The objective is to choose an optimal preferred velocity $v_A{}^*$ which lies as close as possible to $v_A^{pref}$, such that collisions among the robots are avoided for at least a time horizon $\tau$.

The velocity obstacle $VO_{A|B}^{\tau}$ for $\mathcal{R}_A$ induced by $\mathcal{R}_B$ in the local time interval $[0, \tau]$ is the set of velocities of $\mathcal{R}_A$ relative to $\mathcal{R}_B$ that will cause a collision between $\mathcal{R}_A$ and $\mathcal{R}_B$ at some moment before time $\tau$ has elapsed. It is assumed that both robots maintain a constant trajectory within that time interval. Formally:

$$VO_{A|B}^{\tau} = \{v | \exists t \in [0, \tau] :: t(v - v_B) \in D(p_B - p_A, r_A + r_B)\}, \tag{6}$$

where $D(p_B - p_A, r_A + r_B)$ denotes an open disc of radius $(r_A + r_B)$ centered at $(p_B - p_A)$. If $\mathcal{R}_A$ and $\mathcal{R}_B$ each choose a velocity outside $VO_{A|B}^{\tau}$ and $VO_{B|A}^{\tau}$, respectively, then they will be collision-free for at least period of time $\tau$.

The half-plane of velocities $ORCA_{A|B}^{\tau}$ can be constructed geometrically as follows. Let us assume that $\mathcal{R}_A$ and $\mathcal{R}_B$ adopt velocities $v_A$ and $v_B$, respectively, and that these velocities causes $\mathcal{R}_A$ and $\mathcal{R}_B$ to be on collision course, i.e., $v_A - v_B \in VO_{A|B}^{\tau}$. Let $\mathbf{w}$ be the vector from $v_A - v_B$ to the closest point on the boundary of the velocity obstacle:

$$\mathbf{w} = (\text{argmin}\, C_{v \in \partial VO_{A|B}^{\tau}} ||v - (v_A - v_B)||) - (v_A - v_B). \tag{7}$$

Then, $\mathbf{w}$ is the smallest change required to the relative velocity of $\mathcal{R}_A$ and $\mathcal{R}_B$ to avoid a collision within $\tau$ time. To *share responsibility* among the robots, $\mathcal{R}_A$ adapts its velocity by (at least) $(1/2)\mathbf{w}$ and assumes that $\mathcal{R}_B$ takes care of the other half. Hence, the set $ORCA_{A|B}^{\tau}$ of allowed velocities for $\mathcal{R}_A$ is the half-plane pointing in the direction of $\mathbf{n}$ starting at the point $v_A + (1/2)\mathbf{w}$, where $\mathbf{n}$ is the outward normal of $VO_{A|B}^{\tau}$ at $v_A - v_B + \mathbf{w}$ [4]:

$$ORCA_{A|B}^{\tau} = \left\{ v | \left( v - \left( v_A + \frac{1}{2}\mathbf{w} \right) \right) \cdot \mathbf{n} \geq 0 \right\}. \tag{8}$$

The preferred velocity described in the previous section is passed along to the classic ORCA algorithm that sets the new velocity to be applied to the agent. In other words, the velocity computed by Eq. (5) is informed to ORCA as the *preferred velocity*, which is a value used as a reference to determine the actual velocity that will be assigned to the robot during navigation. Consequently, the value belonging to that half-plane that most closely matches the preferred velocity is calculated using linear programming and passed as the new velocity that the robot should take.

## 4. Experiments

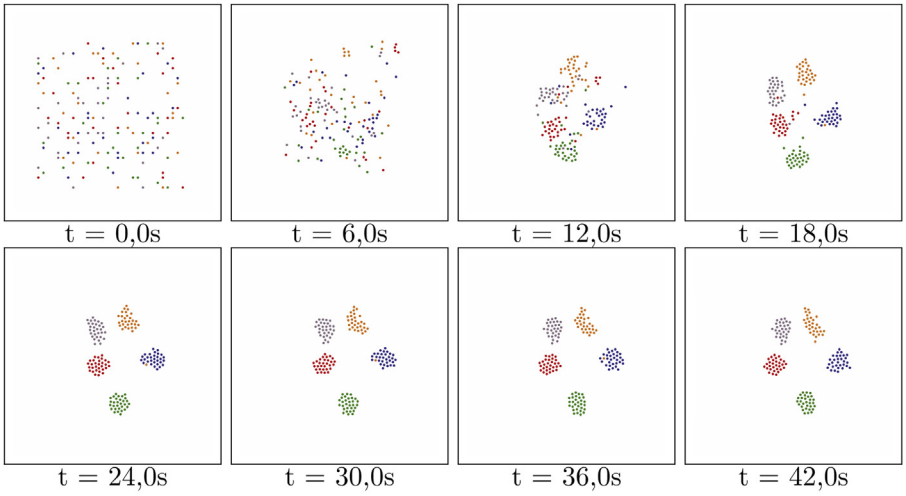
In order to evaluate the proposed algorithm in relation to its performance and feasibility, we performed a series of experiments in simulated scenarios in which we varied the number of groups that form the swarm, the number of agents in each group and scenarios composed by groups of different sizes. Since the proposed technique may produce different results during executions (due to a non-deterministic factor on the velocity selection), we performed 100 simulations to evaluate each set of experiments. A video illustrating the execution is available at https://youtu.be/hSs3m1WtzNk.

Fig. 2 shows snapshots of the execution of the proposed algorithm. In these examples, we have 5, 10 and 15 distinct groups, represented by different colors, each one formed by 30, 15 and 10 agents, respectively. We can see that robots of similar types form small subclusters whose size grows over time, as other agents or subclusters join them. The main advantages of our approach over Santos et al. [2] and Ferreira Filho and Pimenta [15] is the lack of need of global knowledge of all agents positions and a strategy for navigation without collision.

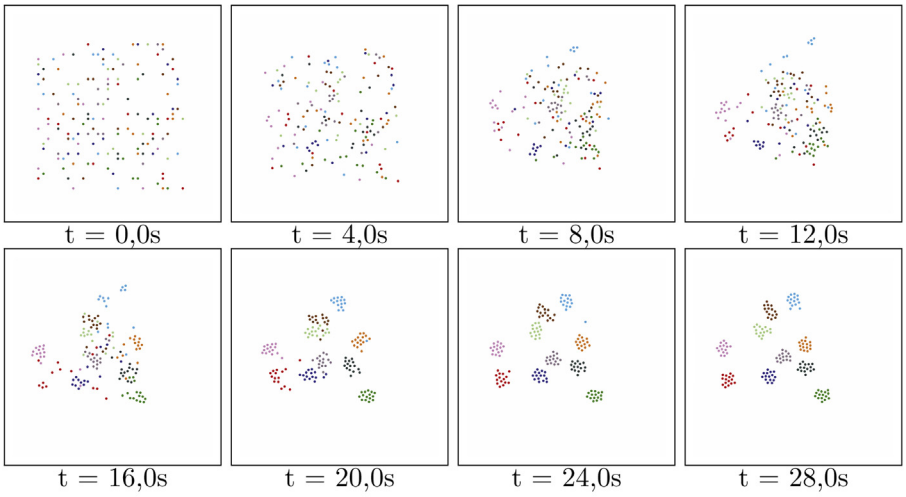
The next experiment was performed in order to investigate the behavior in scenarios with groups of different sizes. In this experiment, we considered a scenario composed of 150 agents distributed into 8 groups of sizes ranging from 8 to 26 agents. Fig. 3 shows snapshots of navigation over time. Differently from Santos et al. [2], it is possible to observe that our proposed algorithm has the same segregative behavior when used in environments with different group sizes.

To better evaluate the segregation, we used a metric proposed in Kumar et al. [14], which consists of calculating the average distances between agents of the same group and agents from different groups.

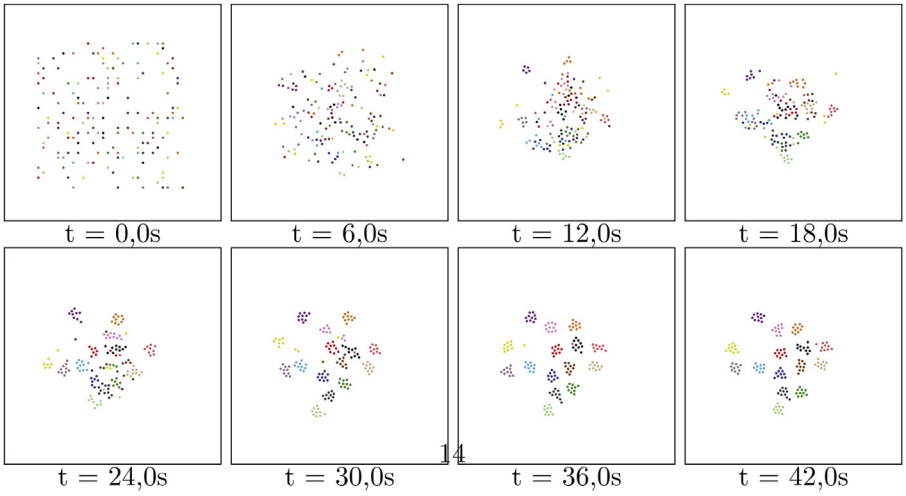
The segregative behavior of the methodology may be confirmed by using the aforementioned metric pairwise. Figs. 4 and 5 depict the average distance between the agents obtained from particular instances carried out with the same initial configuration. As can




(a) 5 groups.




(b) 10 groups.




(c) 15 groups.

**Fig. 2.** Execution of the algorithm in different scenarios composed of 150 agents evenly distributed into 5, 10 and 15 groups.

be seen in these figures, the average distance between agents in the group $A$ are smaller than the average distance between agents from group $A$ relative to agents of all other groups at the end of the execution, i.e., $d_{XX} < d_{XY} \; \forall \; \Gamma_X, \Gamma_Y \in \boldsymbol{R}$. The experiment represented by Fig. 4 was executed without the use of the third component defined in the previous section. On the other hand, Fig. 5 depicts

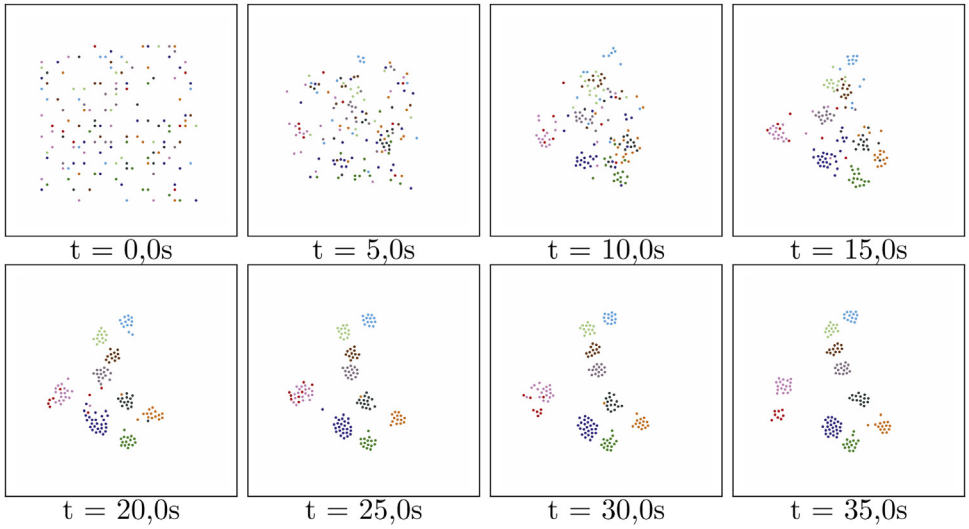



**Fig. 3.** Execution of the algorithm in a scenario with 150 agents distributed in 8 groups of different sizes.

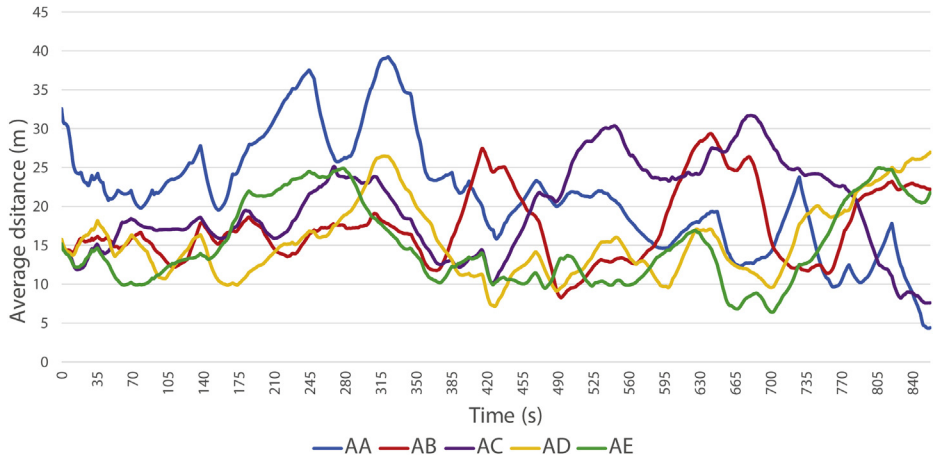



**Fig. 4.** Average distance among agents during the execution of the proposed algorithm without communication between agents.

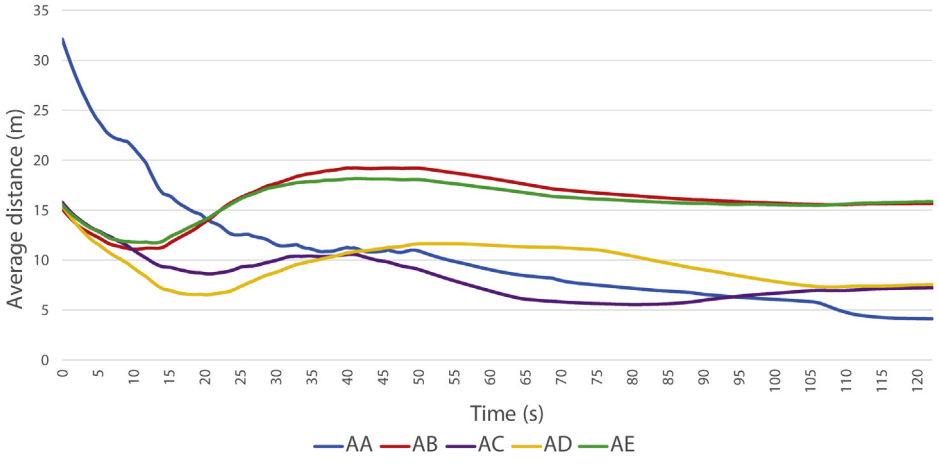



**Fig. 5.** Average distance among agents during the execution of the proposed algorithm with communication between agents.

an experiment in which the information exchange represented by this component was used.

As can be seen in Fig. 4, the subgroups formed during the execution of the task may take a longer time to merge into separate group. Since the task is executed in a very large work environment, considerably larger than the area that an agent can sense, small groups that form at the beginning of the task can move in divergent directions for a long period of time, making it impossible to

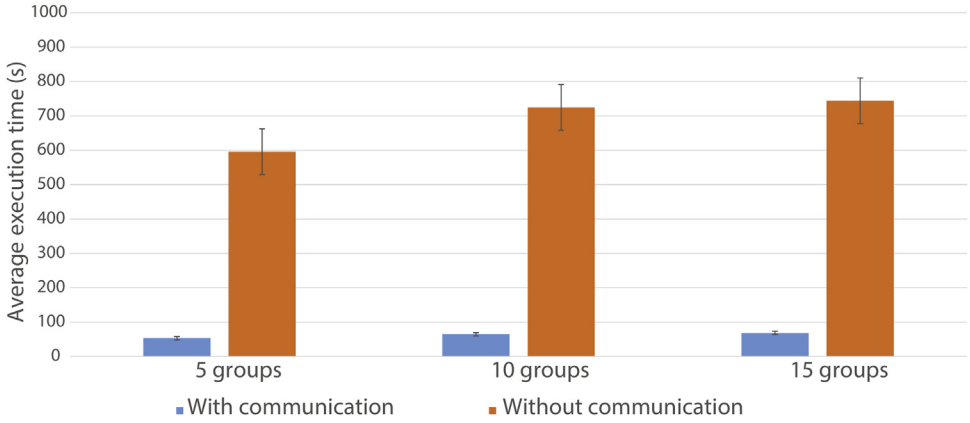



**Fig. 6.** Average execution time of the proposed algorithm with different number of groups and a fixed number of evenly distributed agents per group (150 agents on total).

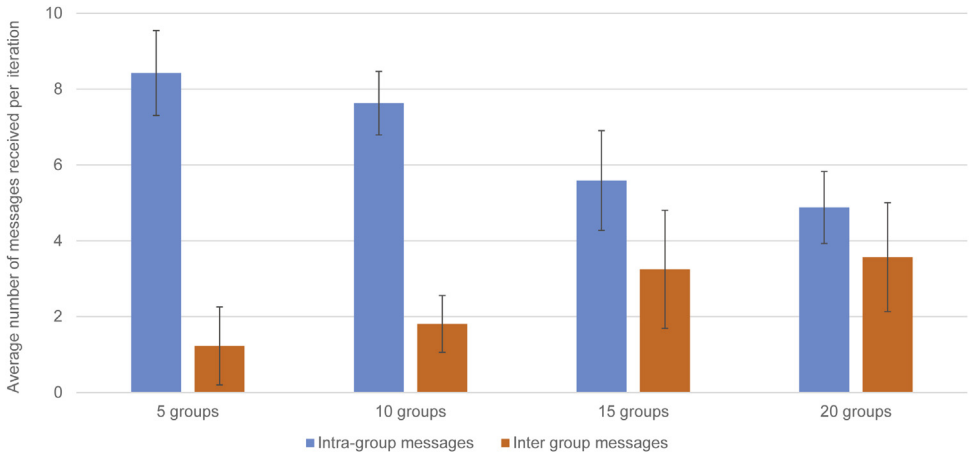



**Fig. 7.** Average number of messages received by an agent during an iteration of the proposed algorithm with different number of groups and a fixed number of evenly distributed agents per group (180 agents on total).

the completion of the task quickly. By introducing the exchange of information between the agents (Fig. 5), these subgroups begin to present a search behavior directed by the other subgroups formed by agents of their own type. Thus, the exchange of information between agents is important to reduce the convergence time of the algorithm.

As mentioned earlier, the first set of experiments took into account the variation in the number of groups that form the swarm. To evaluate the impact of this variation, we used 3 scenarios in which a swarm composed by a total of 150 agents evenly distributed into 5, 10 and 15 groups, respectively. As one can see in Fig. 6, the increase in the number of groups caused a small increase in the time needed to complete the task when communication between the agents was used. In the experiments that did not use communication, the increase presented was more significant. We can also observe that the use of communication between the agents allows a reduction of up to 90% in the total time spent for the execution of the task.

Fig. 6 shows that the use of communication has resulted in a significant improvement in system performance. However, it is important to evaluate the cost of using communication between agents. Fig. 7 shows the average number of messages received by the agents in each iteration of the execution of the experiments performed. Data were stratified into two categories, intra-groups and inter-groups. The intra-group category shows the messages exchanged between agents belonging to the same group. On the other hand, the inter-group category exposes the quantities of messages exchanged between agents from different groups. It is possible to see that the average number of messages exchanged between agents in the same group decreases as the number of groups increases, while inter-group messages increase. This behavior is easily justified by the fact that by increasing the number of groups sharing the same environment, the probability of agents belonging to different groups also increases. It is possible to notice that the average number of messages exchanged between the agents decreases slowly with the increase in the number of groups. Given the large number of agents that share the work environment, we find it advantageous to use less than 10 messages per iteration to achieve results up to 90% better than those found without the use of agent-to-agent communication.

In the second set of experiments, we investigated the performance of the proposed algorithm in relation to the variation in the number of agents in each group. The simulations were performed with 5 groups consisting of 20, 40 and 60 agents. As can be seen in Fig. 8, the increasing in the number of agents per group caused an increase in execution time, a similar behavior to the one observed in the first set of experiments. This increase may be related to the fact that one group is able to “catch” and “drag” a small amount of agents from another group, usually one or two agents, as they navigate. With the increasing in the number of agents and groups in the work environment, the number of interactions of this type increases, which can lead to a degradation of overall system performance.

We have also performed an experiment to compare the proposed approach with the algorithm presented in [2]. We highlight the fact that, unlike what happens in [2], we only use local sens-

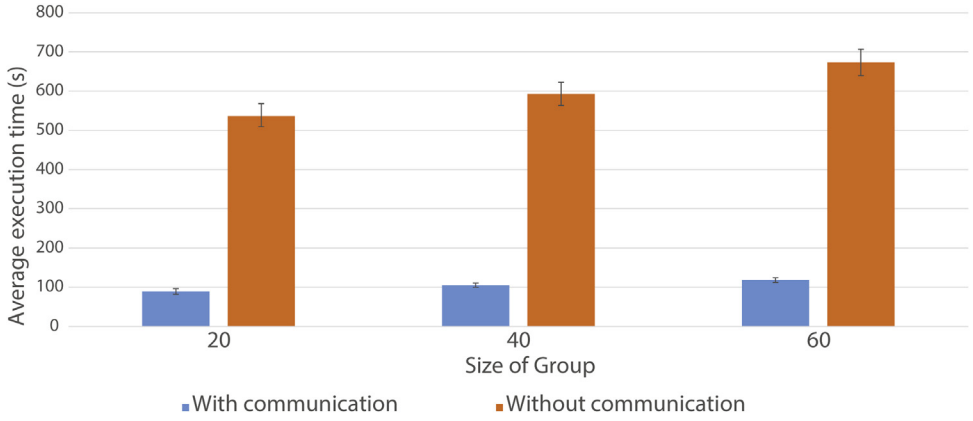



**Fig. 8.** Average execution time of the proposed algorithm with different number of agents per groups and fixed number of groups (5 groups on total).

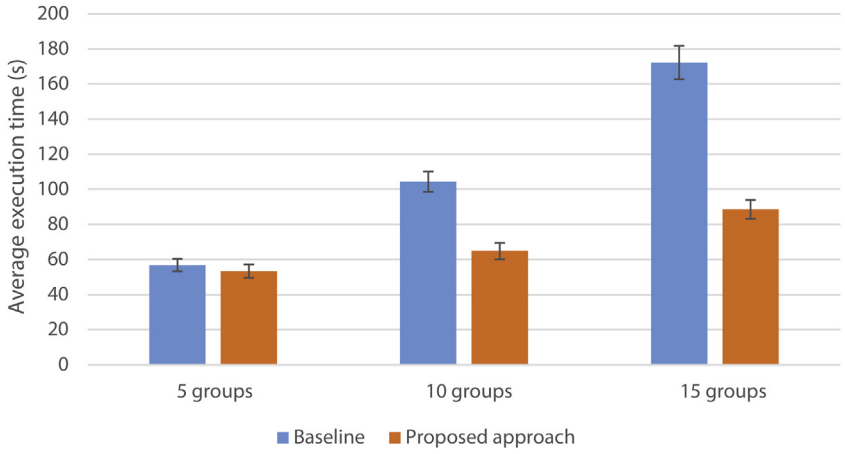



**Fig. 9.** Comparison between the proposed approach and the algorithm developed in [2] with different number of groups and a fixed number of evenly distributed agents per group (150 agents on total).

ing to obtain information about the environment. Nevertheless, the proposed approach overcame the work developed by the authors in all scenarios tested, as it is possible to see in Fig. 9. A possible justification for this is precisely the fact that the baseline algorithm uses a global view, as the agents can perceive the whole environment, the tendency observed is that all the agents move to the center of the environment, which causes an area of congestion that degrades the performance of the system.

## 5. Conclusion

In this work, we proposed a decentralized methodology that can sort a swarm composed of distinct groups of robots, so that agents of the same type remain close to each other, while agents of distinct types remain separated. Our approach consists of extending the ORCA algorithm with a modified version of the PSO method: using the information captured by local sensors, each robot can evaluate its neighborhood in order to calculate a speed that keeps it close to agents of its own group. This velocity is used as the preferred velocity by the ORCA algorithm, which is responsible for navigating the robot while avoiding other agents.

Several experiments were performed to evaluate the proposed methodology and the results showed that our algorithm is an effective alternative to segregate different robots groups on a shared environment. The algorithm was capable of segregating unbalanced groups using only local information, and successfully avoiding collisions with other robots, characteristics that were not present in previous segregation algorithms.

Future research directions include the evaluation of different segregation metrics, such as the intersection area of the convex hull formed by the agents and a possible network connectivity inside the group. We also intend to consider standard metrics of separation used in cluster analysis, for example, the distance between centroids of the groups weighted by their variance. More experiments should also be performed using a larger number of real robots to better analyze the behavior of the algorithm in real scenarios.

## References

[1] E. Şahin, Swarm robotics: from sources of inspiration to domains of application, in: Swarm Robotics, Springer, 2005, pp. 10–20.
[2] V.G. Santos, L.C. Pimenta, L. Chaimowicz, Segregation of multiple heterogeneous units in a robotic swarm, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 1112–1117.
[3] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, MHS'95, IEEE, 1995, pp. 39–43.
[4] J. Van Den Berg, S.J. Guy, M. Lin, D. Manocha, Reciprocal n-body collision avoidance, in: Robotics Research, Springer, 2011, pp. 3–19.
[5] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1986) 90–98.
[6] J. Borenstein, Y. Koren, The vector field histogram – fast obstacle avoidance for mobile robots, IEEE Trans. Robot. Automat. 7 (1991) 278–288.
[7] D. Fox, W. Burgard, S. Thrun, et al., The dynamic window approach to collision avoidance, IEEE Robot. Automat. Mag. 4 (1997) 23–33.
[8] A. Chakravarthy, D. Ghose, Obstacle avoidance in a dynamic environment: a collision cone approach, IEEE Trans. Syst. Man Cybernet. A: Syst. Hum. 28 (1998) 562–574, http://dx.doi.org/10.1109/3468.709600.
[9] P. Fiorini, Z. Shiller, Motion planning in dynamic environments using velocity obstacles, Int. J. Robot. Res. 17 (1998) 760–772.

[10] L. He, J. van den Berg, Meso-scale planning for multi-agent navigation, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2013).
[11] J. Van Den Berg, S.J. Guy, M. Lin, D. Manocha, Optimal Reciprocal Collision Avoidance for multi-agent navigation, in: Proc. of the IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 2010.
[12] D. Wilkie, J. Van den Berg, D. Manocha, Generalized velocity obstacles, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, IROS 2009, IEEE, 2009, pp. 5573–5578.
[13] L.E. Quevillon, E.M. Hanks, S. Bansal, D.P. Hughes, Social, spatial, and temporal organization in a complex insect society, Sci. Rep. 5 (2015) 13393.
[14] M. Kumar, D.P. Garg, V. Kumar, Segregation of heterogeneous units in a swarm of robotic agents, IEEE Trans. Automat. Control 55 (2010) 743–748.
[15] E.B. Ferreira Filho, L.C.A. Pimenta, Segregating multiple groups of heterogeneous units in robot swarms using abstractions, Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS) (2015) 401–406.
[16] R. Groß, S. Magnenat, F. Mondada, Segregation in swarms of mobile robots based on the brazil nut effect, Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS) (2009) 4349–4356.
[17] V.G. Santos, L. Chaimowicz, Cohesion and segregation in swarm navigation, Robotica 32 (2014) 209–223.
[18] C.W. Reynolds, Flocks, herds and schools: a distributed behavioral model, in: ACM SIGGRAPH Computer Graphics, Vol. 21, ACM, 1987, pp. 25–34.
[19] F.R. Inácio, D.G. Macharet, L. Chaimowicz, United we move: decentralized segregated robotic swarm navigation, in: Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics, Springer, 2016.
[20] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inf. Process. Lett. 85 (2003) 317–325.
[21] D. ping Tian, A review of convergence analysis of particle swarm optimization, Int. J. Grid Distrib. Comput. 6 (2013) 117–128.
[22] K.E. Parsopoulos, M.N. Vrahatis, Particle swarm optimization method in multiobjective problems, in: Proceedings of the 2002 ACM Symposium on Applied Computing, ACM, 2002, pp. 603–607.

**Fabrício Rodrigues Inácio** is a Ph.D. student at the Computer Science Department of Universidade Federal de Minas Gerais (UFMG). He received his M.Sc. degree in Computer Science from the same university in 2016. He is with the Computer Vision and Robotics Laboratory (VeRLab), and his main research interests are in mobile robotics, cooperative robotics, coordination of robotic swarms, and digital games.

**Douglas G. Macharet** is an Assistant Professor at the Computer Science Department of Universidade Federal de Minas Gerais (UFMG). He received both M.Sc. and D.Sc. degrees in Computer Science from the same university in 2009 and 2013, respectively. He is with the Computer Vision and Robotics Laboratory (VeRLab), and his main research interests are in mobile robotics, localization, mapping, path planning for non-holonomic robots, multi-robots systems, and human-robot interaction.

**Luiz Chaimowicz** is an Associate Professor at the Computer Science Department of Universidade Federal de Minas Gerais (UFMG). He received his Ph.D. degree in Computer Science from this same university in 2002, and from 2003 to 2004, he held a Postdoctoral Research appointment with the GRASP Laboratory at University of Pennsylvania. He co-directs UFMG's VeRLab and J Lab. His research interests include cooperative robotics, coordination of robotic swarms, and digital games.