2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

# Network Topology Inference in Swarm Robotics

Bruno Luis Mendívez Vásquez and Jan Carlo Barca

*Abstract*— **Swarm robotics refers to the implementation of swarm intelligence features like autonomy and self-organization to a collective of robots. This study focuses on the construction of a topological graph that represents both the magnitude and orientation of swarm interactions. Such structure is used for identifying global parameters like leadership and to derive a relationship between the distribution of interaction magnitudes and swarm parameters. Interaction magnitudes were derived from the trajectory distance between nearest neighbors and it was found that the distribution is able to differentiate between only a small subset of controllers, communication ranges and swarm sizes. Leader detection was based on the analysis of position vectors orientation in local neighborhoods. The method was successful at a 100% rate for 10 and 30 robots, while for 60 a minimum rate of 67% was obtained. Additionally, processing times never exceeded a simulation duration for swarms up to 30 robots, with the potential to parallelize for larger sizes.**

## I. INTRODUCTION

A swarm consists of a collection of individuals that can share the same behavior, thus allowing well established local interactions to occur among the individuals and the surrounding environment. This local set of microscopic behavior is said to trigger a more global set of macroscopic and complex actions that can ultimately converge in achieving a certain predefined goal [1]. In particular, robotic swarms are characterized by their autonomy, decentralized control [3], local interaction and biological inspiration [2].

The core communication procedures in a robotic swarm is implemented through a *controller*, a model that governs the individual behavior of agents in a swarm. A controller defines a behavioral rule set for each agent, so their execution can trigger local interactions among individuals [4]. Controller models get inspiration from particular collective behaviors found in nature [5], [6]; laws of physics [7], [8], [9] and even problem optimization methods [10].

The logical structure that represents the connections among individuals based on information exchange is the *network topology* of a swarm. This topology is dynamic since it changes over time depending on the controller and the convergence to a goal. A single connection in the topology could be seen as an *influence* between two agents; one that was either a direct transmission of information (explicit), or a stimulus that causes the reaction of the other individual (implicit). A graph is a common formalization for a network topology [11].

Bruno Luis Mendívez Vásquez is with Faculty of Information Technology, Monash University, Wellington Rd, Clayton VIC 3800, Australia bruno.mendivezvasquez@monash.edu
Jan Carlo Barca is with Faculty of Information Technology, Monash University, Wellington Rd, Clayton VIC 3800, Australia jan.barca@monash.edu

In order to understand, describe, model and replicate swarm intelligent systems, it is required to *infer* a set of global behavioral parameters, starting from the observation of motion and the subsequent description of dynamic topologies [12]. Furthermore, the goal of this paper is to address the network topology inference problem as the estimation of global behavioral parameters from the reconstruction of a topological graph. The reconstruction process relies on passive observations of motion trajectories, in order to identify and quantify local interactions without interfering with the normal development of the swarm's behavior.

Inference approaches are most often based on *system identification* methods that compute a mathematical model of a swarm tuned by a set of parameters [13]. The output data of the model is then compared to the ground truth to estimate an error function [14]. Eriksson *et al.* [15] conducted a study to reconstruct local interaction rules from a simulated model of a swarm represented by the acceleration vectors of each agent. The method was used to extract interaction features of the individuals such as the degree of collision avoidance, alignment with local speed, and movement to local average position. The work of [15] is a theoretical introduction to interaction rules inference, heavily dependent on the tested controller. A similar approach was proposed by Lukeman *et al.* [16], but this time the project aimed to reconstruct positions, velocities and trajectories from the observed data to generate spatial and angular neighbor distribution plots. The range of interaction for an individual is divided into repulsion, alignment and attraction zones [16]. The main advantage of this model is the visual representation of collective motion insights, feature that inspired this study to build a heat map of interactions for a network topology.

Probabilistic modeling and a Bayesian inference method was introduced by Mann [17]. It is based on determining the probability distribution of parameter values when taking the observed swarm data as input, and considering a *likelihood* function as the probability distribution of a set of observations that focus on orientation changes in particles. Following a similar probabilistic approach, Correll & Martinoli [18] shows how almost any aspect of interest can be modeled by a simple metric-based predictive model where a set of predefined parameters can be optimized by the least squares method. Unlike this study, the approach for system identification is controller centered and it focuses on a single aspect of swarm behavior at a time (e.g. collision avoidance).

Kunz [19] developed a deterministic method to reconstruct a topological graph based on the identification of neighborhoods and the correlation of velocity and acceleration fluctuations among agents. Global parameters considered by [19] are the communication range of each individual and leaders. A disadvantage of using this method is the high entropy required to rely on useful correlations and thus link detections. Static formations are not well supported unless a high amount of perturbations is considered, the opposite of relying on passive observations which is a feature implemented in this paper. One major consideration when tuning the detection methods is the use of thresholds for each criterion. A similar approach is proposed by Nickson [20] where an interaction is defined as the product of the acceleration of two robots. In [20], interactions are filtered by focusing on strong accelerations which is hypothesized to be caused by a response to communication. In contrast, this study quantifies an interaction as the distance between two trajectories. For leader detection, the main hypothesis behind [20] is that velocity vectors in leaders have an additional component influenced by the position of the goal, while followers do not have such a component, just the influence of neighboring robots. Unlike [20], a more geometric approach is considered in this paper by analyzing the orientation of nearest neighbors.

The following contributions are the result of the work presented in this paper:

- The reconstruction of a topological graph is based on passive observations of motion trajectories. No perturbations are needed to infer swarm parameters.
- The topological graph is considered to be weighted so interaction magnitudes are recorded. Orientations are also considered, so leadership magnitudes can be calculated for every node.
- Interactions are formalized as trajectory distances between two nodes.
- Global swarm parameters are inferred by analyzing the extracted topological graph. As seen in [19] and [20], the swarm communication range and leaders are also considered to be the main inferred swarm parameters.

## II. NETWORK TOPOLOGY INFERENCE

Formally, a swarm $\mathcal{S}$ under controller $C$ is a set of $n$ particles $p_i, i = 1, \ldots, n$ whose position is a function of time $\mathbf{r}_i(t) \in \mathbb{R}^3$, $t = 1, \ldots, m$ where $m$ is the number of time steps for the execution of $\mathcal{S}$. The controller $C$ updates the position of an individual by combining the velocity vectors of nearest neighbors in a communication range.

A *topological graph* in time step $t$ is a weighted and directed graph $G(t) = (V, E)$ where $p_i \in V$ (particle set of $\mathcal{S}$) and $e = (p_i, p_j) \in E$ is an edge with orientation $p_i \to p_j$ if $p_i$ is *following* $p_j$ [20]. However, not every edge includes orientation information about the particles since it is arbitrarily assumed that the *following* relationship is one-to-one, although one particle can have many followers. A *weight* function $w(e) \in \mathbb{R}^3$ measures the interaction magnitude between two particles. It is a scalar value proportional to their trajectories distance based on position.

A collection of magnitudes and orientations from the topological graph can be analyzed to infer *global behavioral parameters* of $\mathcal{S}$ such as the communication range $\rho$ and leadership function $\lambda$. The parameter $\rho$ indicates the sphere's radius that a controller $C$ uses to delimit the influence of a particle's neighborhood. Finally, $\lambda(p_i)$ is the *leadership magnitude* for particle $p_i$.

The current approach is described as follows: based on the changing positions of swarm individuals, a Delaunay *triangulation* is computed for every time step. Edges are *tracked* and grouped by the time frames they exist so for every pair of robots that conform an edge, their trajectories are matched for the corresponding time frame. Such *trajectory distance* constitutes the interaction magnitude between two individuals and the weight of every edge in the topological graph. Finally, directions are calculated by inferring *followers* in local neighborhoods.

### A. Triangulation

A particle's *neighborhood* can be derived from an initial *triangulation* of $\mathcal{S}$. Since a proper representation of locality is desired, a triangulation that preserves short distance connections uniformly along the mesh should be considered. The *Delaunay triangulation* addresses previous locality constraints by only generating triangles whose circumscribed circles are empty. This is called the *empty circle property*.

According to Maus & Drange [21], any closest neighbor $q_j$ to a particle $p$ are connected by Delaunay segments $pq_j$ as long as circumscribed circles with diameter $pq_j$ remain empty. Since every Delaunay segment has an empty circumscribed circle by definition, all particles connected to $p$ are closest neighbors. Consequently, the Delaunay triangulation of $\mathcal{S}$ is the undirected graph $D(t) = (V', E')$ of particles and edges where the *neighborhood* of a particle $p_i$ at time step $t$ is defined as $N_i(t) = \{p_j \in V' | (p_i, p_j) \in E'\}$.

### B. Edge Tracking

Edges of the triangulation are constantly changing, some might be lost after a certain period and others preserved throughout all time steps. *Edge tracking* refers to the mapping of every unique edge $e$ in $D(t), \forall t$ to a time range $[t_s, t_e]$, such that $e \in E'$ of $D(t)$, where $t = t_s, \ldots, t_e$. This process outputs a set $\mathcal{W}$ such that, for each edge $e \in \mathcal{W}$, there is a queue $\tau(e)$ whose elements are tuples $(t_s, t_e)$ that record all instances of $e$ in the triangulation. Consequently, an edge can exist in multiples ranges, therefore the use of a queue. The pseudocode of the tracking process that calculates $\mathcal{W}$ from $D(t)$ is shown in Algorithm 1.

For each time step $t$ and particle $p_i$, the neighborhood $N_i(t)$ of $p_i$ is retrieved. With every neighbor $q$, the edge $e = (p_i, q)$ is tracked, with duplicated edges $(q, p_i)$ being ignored, and verified if it is an element of $\mathcal{W}$; if it is not, the edge is marked as existing in the current time step $t$, thus adding the tuple $(t, t)$ to the $\tau(e)$ queue and finally $e$ to $\mathcal{W}$. On the other hand, if the edge already exists in $\mathcal{W}$ and its last existence record marks $t_e$ as the previous time step, then that tuple is updated so now the edge finishes at the current time step ($t_e = t$). Another possible scenario is that $e \in \mathcal{W}$ but the tail tuple does not end in the last time step. This indicates that

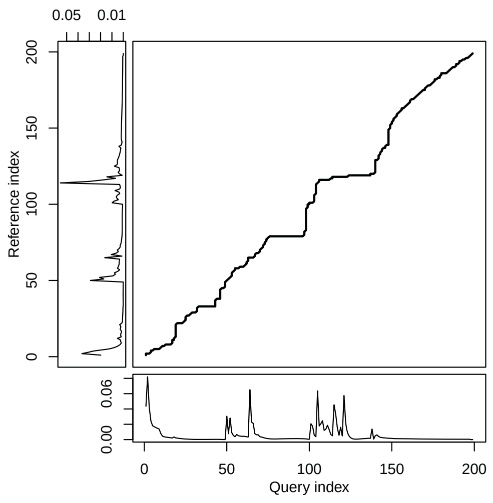


Fig. 1: Warping curve $\phi(k)$ for two position trajectories of particles $p_i$ and $p_j$ (reference and query) conforming edge $e$ during some period $(t_s, t_e) \in \tau(e)$. A perfect match is represented by a positive diagonal (*i.e.* no warping).

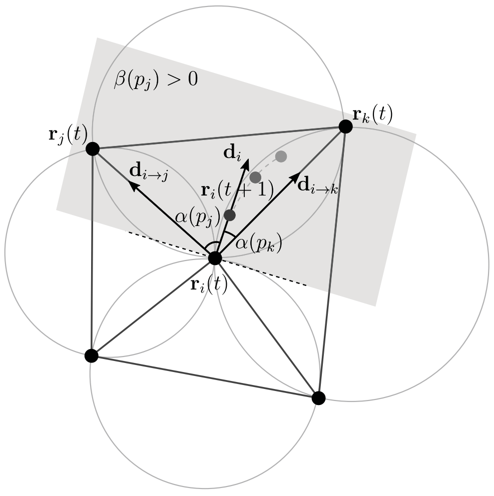


Fig. 2: The neighborhood of $p_i$ is analyzed to determine which particle it follows. Both $p_j$ and $p_k$ are candidates according to the dot product criteria; however, $\alpha(p_k) < \alpha(p_j)$, therefore $p_i$ is following $p_k$ at time step $t$.

**Algorithm 1:** Edge tracking procedure

```
input  : Delaunay triangulation $D(t)$ of $\mathcal{S}$
output: List of edges $e \in \mathcal{W}$ with time ranges $\tau(e)$

for each time step $t$ and particle $p_i$ do
    for each neighbor $q \in N_i(t)$ do
        if $(q, p_i) \notin \mathcal{W}$ then
            $e \leftarrow (p_i, q)$;
            if $e \in \mathcal{W}$ and $t_e = t - 1$ then
                $(t_s, t_e) \leftarrow$ Tail($\tau(e)$);
                if $t_e = t - 1$ then Tail($\tau(e)$) is
                  $(t_s, t)$;
                else add $(t, t)$ to $\tau(e)$;
            else
                add $(t, t)$ to $\tau(e)$;
                add $e$ to $\mathcal{W}$;
```

$e$ is appearing again in the triangulation, after not existing in previous time steps. In this case, a new tuple $(t, t)$ is added to the queue $\tau(e)$. At the end of the procedure, the set $\mathcal{W}$ conforms all unique edges of the triangulation together with their existence periods.

## C. Trajectory distance

Pairs of nearest neighbors that form edges with specific time lengths are suitable for analysis on interaction measures. This analysis is based on the trajectories of both particles, represented as *time series* of their positions in $\mathbb{R}^3$. The distance measure for a pair of trajectories indicates a proportional measure of interaction between the two.

Given an edge $e = (p_i, q) \in \mathcal{W}$ with a time length represented by the tuple $(t_s, t_e) \in \tau(e)$. The *trajectory* of $p_i$, delimited by one time frame of $e$ and based on the position vector $\mathbf{r}$, is the vector $\Gamma_{i,t_s \to t_e}$ such that:

$$\Gamma_{i,t_s \to t_e} = (\mathbf{r}_i(t_s), \mathbf{r}_i(t_s + 1), \ldots, \mathbf{r}_i(t_e - 1), \mathbf{r}_i(t_e)) \quad (1)$$

To derive a distance between two trajectories, the series are compared so a difference in shape can be quantified. The Dynamic Time Warping (DTW) algorithm accomplishes this by stretching and compressing two time series at every time step so one can resemble the other as much as possible [22]. After altering the signals, the sum of distances between aligned points is the total distance between the two series.

To calculate the distance between two single-variate time series $X = (x_i, x_{i+1}, \ldots), i \in [1, N]$ and $Y = (y_j, y_{j+1}, \ldots), j \in [1, M]$, DTW only takes as main input a *cross-distance matrix* $d(i, j) = f(x_i, y_j) \geq 0$ where $f$ is a *local dissimilarity function* that indicates the distance between two points (*e.g.* Euclidean, although a custom one can be specified). Since $f$ is defined arbitrarily $x_i$ and $y_j$ can be scalars or vectors (allowing multivariate series).

The final distance is obtained by calculating the *warping curve* $\phi(k) = (\phi_x(k), \phi_y(k))$, where $\phi_x(k) \in [1, N]$ and $\phi_y(k) \in [1, M]$. Also, and in order to avoid loops, $\phi_{x,y}(k + 1) \geq \phi_{x,y}(k)$. The warping curve is the remapping of time indexes of the signals being compared so they can resemble each other as close as possible. The curve then generates two warped time series whose distortion is then compared. The path that generates the minimum distortion is the final warping path $\phi$ with the final distance being the accumulative distortion between the warped series [22].

In the case of deriving an interaction magnitude from a pair of connected particles, the goal is to obtain a normalized distance measure between the series $X = \Gamma_{i,t_s \to t_e}$ and $Y = \Gamma_{j,t_s \to t_e}$ corresponding to the trajectories of particles $p_i$ and $p_j$ such that $e = (p_i, p_j) \in \mathcal{W}$ and $(t_s, t_e) \in \tau(e)$. Both trajectories are multivariate series since their elements are position vectors $\mathbf{r} \in \mathbb{R}^3$. In such case, the dissimilarity function $f$ for the cross-distance matrix $d$ is still Euclidean [22]:

$$d(u,v)^2 = \sum_{\delta=1}^{3} (X_{u,\delta} - Y_{v,\delta})^2 \tag{2}$$

In Equation 2, both series $X$ and $Y$ have the same length so $u, v \in [1, N]$ and $N = t_e - t_s + 1$. Also, the dimension $\delta$ refers to the $x, y$ or $z$ component of $\mathbf{r}$ so, for instance, $X_{2,3} = r_z(t_s + 1)$ of particle $p_i$.

By applying DTW to $X$ and $Y$ for every edge $e$ with length of time frame $(t_s, t_e)$ greater than 1, the resulting warping curve $\phi(k)$ can be utilized to obtain the normalized distance $\omega_{e,t_s \to t_e}(u) \in [0, 1]$, where $u \in [1, N]$. This vector indicates the difference between the warping curve $\phi(k)$ and the diagonal that represents a perfect match (Fig. 1) for every $u$ in the current time frame of $e$. The pseudocode of the process that obtains the distance between two trajectories for every edge in $\mathcal{W}$ is shown in Algorithm 2.

**Algorithm 2:** Trajectory distance calculation

```
input : List of edges $e \in \mathcal{W}$ with
        $t_e > t_s, \forall (t_s, t_e) \in \tau(e)$
output: Normalized distance vector $\omega_{e,t_s \to t_e}(u)$ for
        every input

for each edge $e = (p_i, p_j)$ and time frame
 $(t_s, t_e) \in \tau(e)$ do
    $X \leftarrow \Gamma_{i,t_s \to t_e}$;
    $Y \leftarrow \Gamma_{j,t_s \to t_e}$;
    $\phi(k) \leftarrow$ DTW $(X, Y)$;
    for $u \leftarrow 1$ to $N \leftarrow t_e - t_s + 1$ do
        $P \leftarrow \{k | \phi_x(k) = u\}$;
        $h \leftarrow \phi_y($Tail$(P)) - u$;
        $\omega_{e,t_s \to t_e}(u) \leftarrow |1 - \frac{h}{N-1}|$;
```

The input for the algorithm is every edge in $\mathcal{W}$ with a time frame length greater than one. For every edge and corresponding time frames, two particle trajectories are obtained. Then, the Dynamic Time Warping algorithm is applied to them so a warping curve $\phi(k)$ is retrieved as a result. For every time step $u$ in the current frame $(t_s, t_e)$, indexes of $\phi_x(k)$ corresponding to points occurring at $u$ are retrieved so the furthest $y$ coordinate in $\phi_y(k)$ is compared back to $u$ (coordinate in the perfect match diagonal). The difference is then normalized so the value varies between 0 and 1. As a result, a vector $\omega(u)$ is finally obtained for every time frame of $e \in \mathcal{W}$.

### *D. Leader Detection*

A particle $p_i$ is said to be *following* a particle $p_j$ in a time step $t$ if; first, $p_j \in N_i(t)$ (neighborhood of $p_i$); second, $\mathbf{r}_j(t)$ is inside the half-plane defined by the normal vector $\mathbf{r}_i(t+1) - \mathbf{r}_i(t)$ (difference between positions of $p_i$ at $t$ and $t+1$); and third, the angle between the previous normal vector and $\mathbf{r}_j(t) - \mathbf{r}_i(t)$ (difference between position of $p_j$ and $p_i$ at time step $t$) is minimal. These conditions were inspired by the work of Andersson *et al.* [23] on the detection of leaders and followers from moving points trajectories. Fig. 2 depicts the *follower* relationship. The premise is that a particle is always following the one ahead of it which aligns with a minimum angle. Based on these notions, the process that obtains particles' followers for every time step is shown in Algorithm 3.

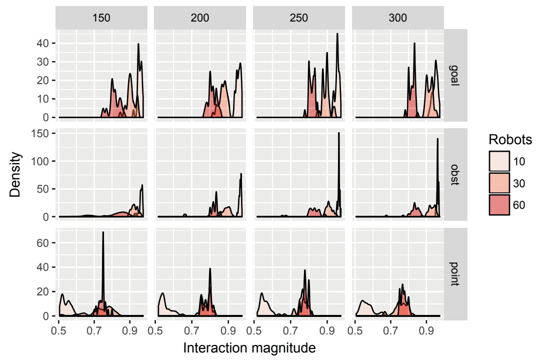
**Algorithm 3:** Procedure to get followers

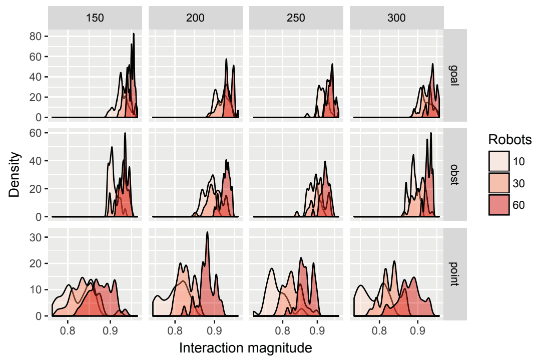
```
input : Delaunay triangulation $D(t)$ of $\mathcal{S}$
output: $p_i$ is following $\psi_i(t) = p_j$ at time step $t$,
        $\forall p_i, t$

for each time step $t$ and particle $p_i$ do
    $\mathbf{d}_i \leftarrow \mathbf{r}_i(t+1) - \mathbf{r}_i(t)$;
    $\alpha_i \leftarrow \operatorname{atan2}(\mathbf{d}_i)$;
    for each neighbor $p_j \in N_i(t)$ do
        $\mathbf{d}_{i \to j} \leftarrow \mathbf{r}_j(t) - \mathbf{r}_i(t)$;
        $\alpha_{i \to j} \leftarrow \operatorname{atan2}(\mathbf{d}_{i \to j})$;
        $\alpha(p_j) \leftarrow |\alpha_i - \alpha_{i \to j}|$;
        $\beta(p_j) \leftarrow \mathbf{d}_i \cdot \mathbf{d}_{i \to j}$;
    $\psi_i(t) \leftarrow \min_{p_j} \{\alpha(p_j) | \beta(p_j) > 0, \forall p_j \in N_i(t)\}$;
```

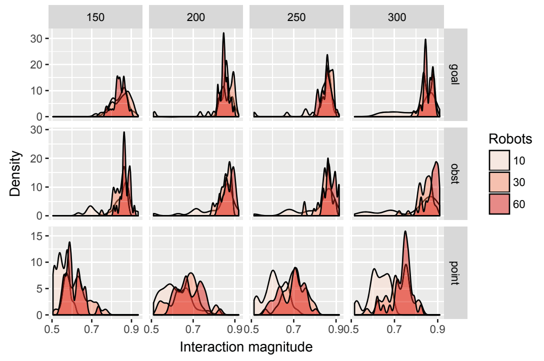
For every particle $p_i$ and time step $t$, the orientation vector $\mathbf{d}_i$ of $p_i$ relative to its position at $t$ and $t+1$ is obtained. For every neighbor $p_j$ of $p_i$ according to the triangulation, an orientation vector $\mathbf{d}_{i \to j}$ starting from the position of $p_i$ to $p_j$ is calculated. The angle between these two vectors $\alpha(p_j)$ is kept for every neighbor $p_j$, as well as the dot product $\beta(p_j)$. With the $\alpha$ and $\beta$ vectors that apply to the neighborhood of $p_i$, the particle that $p_i$ is following at $t$, $\psi_i(t)$, is calculated as the $p_j$ inside the half-plane defined by the normal $\mathbf{d}_i$ (*i.e.* dot product $\beta(p_j) > 0$), and also whose angle with $\mathbf{d}_i$ is minimal. If none of these conditions apply, then $p_i$ is not following any particle ($\psi_i(t) \leftarrow \varnothing$).

In order to identify particles as followers based on $\psi_i(t)$, a *follower chain* is derived per time step and particle. This chain initiates at $p_i$ and extends until a particle is found such that $\psi_k(t) = \varnothing$, where $p_k$ is the last particle of the chain. By calculating all possible chains for a time step $t$, a *leadership order* can be inferred by looking at the position of particles in the chains. Particles occurring at the head of a chain are most likely to be leading. The procedure that extracts these chains based on $\psi_i(t)$ is shown in Algorithm 4.

For every particle $p_i$ and time step $t$, a chain $\pi_i(t)$ is a sequence of tuples of the form $\{(j, c_1), (j-1, c_2), \cdots, (1, c_j)\}$ starting at $c_1 = p_i$, where $j$ is the length of the chain and $c$ values are particles. The sequence depicts a follower chain $c_1 \to c_2 \to \cdots \to c_j$ where every element has a *leadership order* (or position in the chain), starting from 1 for the last element $c_j$, to $j$ for the initial particle $c_1 = p_i$. The particle $p_k$ (initially equal to $p_i$) and subsequent ones are added to the chain until $\psi_k(t) = \varnothing$ or a cycle is detected (*i.e.* $\psi_k(t)$ is already in the chain). Only chains with length greater than one are considered for $\pi_i(t)$. To identify potential leaders, it is possible to sort particles based on their *leadership order* and how often this order repeats throughout time steps.


(a) *Flocking* controller.


(b) *Morphogenesis* controller.


(c) *Physicomimetics* controller.

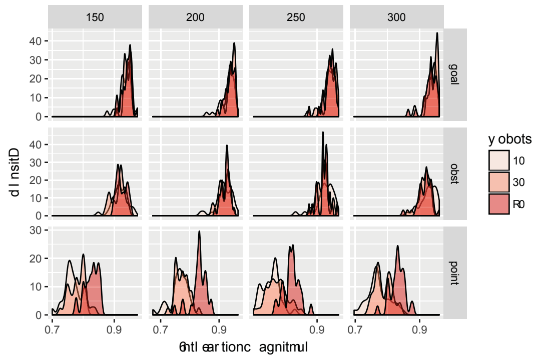

(d) *PSO* controller.

Fig. 3: Interaction magnitude density distribution for different controllers, scenarios, swarm sizes and communication ranges. All edges from the topological graph are considered for the distribution plot. The different shapes of the distributions are visualized as an attempt to derive a relationship between the interaction magnitude of the topological graph and different swarm parameters.

```
Algorithm 4: Procedure to extract follower chains
input : Follower pairs $p_i \to \psi_i(t), \forall p_i, t$
output: Follower chain $\pi_i(t)$, $\forall p_i, t$
for each time step $t$ and particle $p_i$ do
    $p_k \leftarrow p_i$;
    $j \leftarrow 1$;
    $C \leftarrow \{(j, p_k)\}$;
    while $\psi_k(t) \neq \varnothing$ and $\psi_k(t) \notin C$ do
        $j \leftarrow j + 1$;
        $C \leftarrow C \cup \{(j, \psi_k(t))\}$;
        $p_k \leftarrow \psi_k(t)$;
    if $j > 1$ then
        let $C = \{(1, c_1), (2, c_2), \cdots, (j, c_j)\}$;
        $\pi_i(t) \leftarrow \{(j, c_1), (j-1, c_2), \cdots, (1, c_j)\}$;
```

The proposed hypothesis to leader detection assumes that particles in leading positions occur more frequently, thus considering the maximum count of occurrences per particle as a *leadership magnitude*.

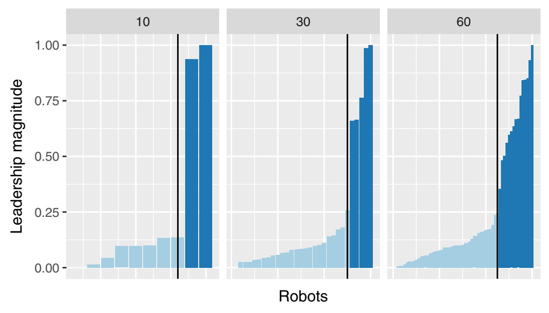
$$\gamma(k, p_i) = \text{count}\{(k, p_i) \in \pi_i(t), \forall t\}$$
$$\lambda(p_i) = \max\{\gamma(k, p_i), \forall k\} \tag{3}$$

In Equation 3, the function $\gamma(k, p_i)$ is the count of all tuples $(k, p_i)$ that occur in $\pi_i(t)$ for all time steps $t$. It is the number of times a particle $p_i$ at position $k$ is found in all chains. The *leadership* function $\lambda(p_i)$ is then the maximum of these occurrences for $p_i$ and all $k$. Sorting $\lambda(p_i)$ for all $p_i$ in descending order gives the distribution of leadership magnitudes that can be used to divide the swarm into leaders and followers. If the number of leaders $L$ is known, then the first $L$ elements of the sorted list lead the swarm.

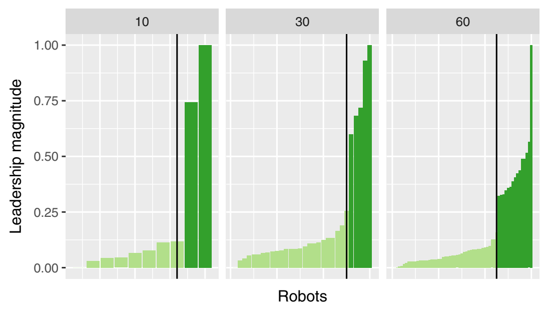
## III. SIMULATIONS

The software used for simulating a swarm is ARGoS [24] where each controller is coded in C++. The hardware platform used for the experiments consisted of an Intel® Core$^{\text{TM}}$ i5-2450M processor running at 2.5 GHz and 4 GB of RAM. The current approach was implemented in R. Experiments are carried out for the following controllers: *flocking* [5], *morphogenesis* [6], *physicomimetics* [9] and *PSO* [10]. For leader detection both *flocking* and *physicomimetics* controllers are used. Leaders influence followers by communicating the goal position so it becomes a component of their velocity vectors which update their positions. Leaders specified by the ground truth conform 20% of the size of the swarm. Robots will be moving inside a 120x120 arena, placed under different swarm sizes and scenarios. The three main ones considered for this research are inspired by the work of Nickson [20]:

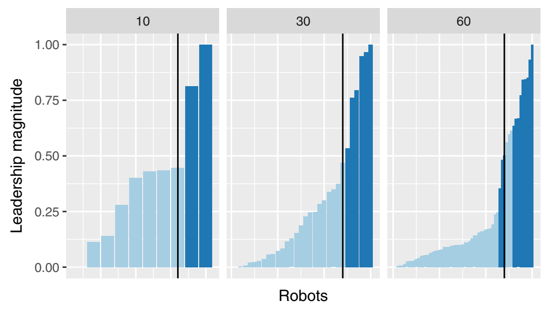
- *Swarming around a point*. For both flocking and physicomimetics controllers, this scenario initializes robots



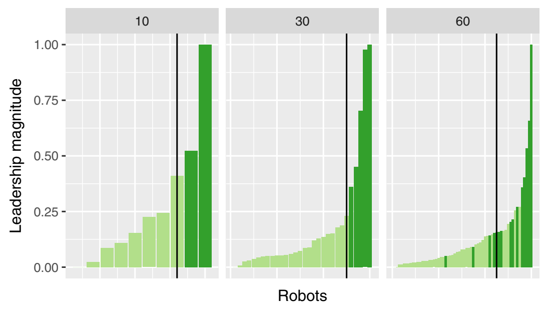
(a) *Flocking* under *goal* scenario.



(b) *Flocking* under *goal with obstacles* scenario.



(c) *Physicomimetics* under *goal* scenario.



(d) *Physicomimetics* under *goal with obstacles* scenario.

Fig. 4: Average leadership magnitude $\lambda$ of all experimental instances for each robot $p_i \in \mathcal{S}$ and different swarm sizes. Values are sorted in ascending order. The vertical line delimits detected leaders. Darker bars are leaders as specified from the ground truth.

around a given point with no goal information. The swarm is stationary with minor movements around its center of mass. For PSO, this is achieved by a minimal fitness function around the initial point. In a morphogenesis swarm, robots are initialized around a point and set a formation.

- *Swarming towards a goal*. Goal information is transmitted to leader robots who then move towards a goal position (by following a light source in the arena). The rest of the swarm follows. When the goal changes, the new position is re-transmitted to the leaders. For a morphogenesis swarm, the formation moves towards the goal without breaking its structure.
- *Swarming through obstacles towards a goal*. Same procedure as swarming towards a goal but in this case, there are static obstacles (robots) aligned in a certain position. The swarm then attempts to move through these obstacles and reach the goal.

Each controller is simulated under the three different scenarios described above. The standard simulation time is 20 s, with a time step set to 10 ms, thus getting 200 time steps in total for an experiment or *instance*. Additional to a scenario and controller, an instance is executed for a specific swarm size (10, 30 or 60) and communication range (150, 200, 250 and 300 cm).

For all controllers and scenarios, the calculation and analysis of a swarm topological graph within an instance execution time is possible under 60 robots. For 30 robots, the maximum processing time of a 20 s instance was 18 s; however, for 60 the time increased to 41 s. It only takes 4 to 5 seconds to process an instance of 10 robots.

## A. Flocking Controller

During swarming around a point (Fig. 3a), the interaction magnitude density increases with the swarm size. It is the opposite for goal-based scenarios where the communication range increases with the interaction density but only for ranges lower than 300. Leader detection (Fig. 4a and 4b) shows an accuracy of 100% for all swarm sizes and after swarm convergence with 200 instances for both goal-based scenarios.

## B. Morphogenesis Controller

The interaction magnitude density increases with the swarm size for all scenarios (Fig. 3b); however, during swarming around a point the peak density is lower than goal-based scenarios. The other two sizes for all scenarios. The communication range did not have a clear influence on the distribution of interaction magnitudes. The achieved formation has a clear structure that individuals gradually build. As the convergence increases, the area of interaction decreases until it remains fixed, independently of the global communication set for the experiment.

## C. Physicomimetics Controller

For the swarming around a point scenario (Fig. 3c), the interaction magnitude increases with the swarm size and also the communication range although subtlety. Goal-based scenarios maintain a solid distribution above 0.8 no matter the swarm size. Communication range in these cases seems to narrow the distribution to higher interaction values. Leader detection for this controller has an accuracy of 100% for 10 and 30 robots in both goal-based scenarios. With no obstacles and 60 robots, the accuracy drops to 75% (Fig. 4c) and 67% (Fig. 4d) when considering obstacles. It is possible that as the swarm increases its size or number of leaders, some will stay in non-leading positions for extended periods of time, perhaps leading the swarm in a more local way, guiding followers at the back of the swarm. Their function has not changed but, as the swarm grows, there might exist logical fractures in the collective that forces individuals to form subgroups with leaders distributed throughout the global swarm structure.

### D. PSO Controller

The PSO controller results in Fig. 3d show very similar distributions for most cases. Swarming around a point shows an increase of interaction magnitude with the swarm size; however, the communication range does not seem to have an influence. For the goal-based scenarios, the interaction is quite similar for all sizes, perhaps narrowing the distribution to higher interaction values as the communication range increases. This only applies when obstacles are not considered though, which is expected.

## IV. CONCLUSION

The experiments demonstrated that swarm global behavioral parameters like leadership can be estimated from the analysis of a edge orientations in a topological graph. Interaction magnitudes were analyzed from a density distribution point of view, offering visual insights about a potential relationship with the communication range. It was found that this global parameter did not have a significant influence on the density distribution for all controllers; however, other parameters like swarm size and scenario performed better. Further classification tasks can be applied to interactions, so more *metadata* could be gathered to support a better identification of global parameters. In addition, scaling compromises leader detection by fragmenting swarms in leadership clusters that require a deeper search for guiding individuals operating in lower hierarchies. Real time processing is also affected by scaling, currently covering only swarm sizes of 30 robots. A more decentralized processing is required with shorter instances being processed in parallel.

## ACKNOWLEDGMENT

The authors would like to thank Clément Kunz (École Polytechnique Fédérale de Lausanne) and Edward Nickson (Monash University) for their previous contributions to this study, and their willingness to share resources and key findings.

## REFERENCES

[1] N. El Zoghby, V. Loscri, E. Natalizio, and V. Cherfaoui, "Robot Cooperation and Swarm Intelligence," in *Wireless Sensor and Robot Networks: From Topology Control to Communication Aspects*. World Scientific Publishing Company, Mar. 2014, pp. 168–201. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00917542

[2] Y. Mohan and S. G. Ponnambalam, "An extensive review of research in swarm robotics," in *World congress on Nature Biologically Inspired Computing*, Dec 2009, pp. 140–145.

[3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[4] E. Şahin, *Swarm Robotics: From Sources of Inspiration to Domains of Application*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 10–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30552-1_2

[5] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Aug. 1987.

[6] Y. Meng, H. Guo, and Y. Jin, "A morphogenetic approach to flexible and robust shape formation for swarm robotic systems," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 25 – 38, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889012001637

[7] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Phys. Rev. Lett.*, vol. 75, pp. 1226–1229, Aug 1995.

[8] D. Strömbom, "Collective motion from local attraction," *Journal of Theoretical Biology*, vol. 283, no. 1, pp. 145 – 151, 2011.

[9] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, March 2006.

[10] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, Oct 1995, pp. 39–43.

[11] A. McNabb, M. Gardner, and K. Seppi, "An exploration of topologies and communication in large particle swarms," in *2009 IEEE Congress on Evolutionary Computation*, May 2009, pp. 712–719.

[12] A. G. Diukman, "Swarm observations: Implementing integration theory to understand an opponent swarm," Master's thesis, Naval Postgraduate School, Monterey, California, United States, 2012.

[13] T. Mostajabi and J. Poshtan, "Control and system identification via swarm and evolutionary algorithms," *International Journal of Scientific and Engineering Research*, vol. 2, no. 10, 2011.

[14] B. Luitel, "Applications of swarm, evolutionary and quantum algorithms in system identification and digital filter design," Master's thesis, Missouri University of Science and Technology, Rolla, Missouri, United States, 2009.

[15] A. Eriksson, M. Nilsson Jacobi, J. Nyström, and K. Tunstrøm, "Determining interaction rules in animal swarms," *Behavioral Ecology*, vol. 21, no. 5, p. 1106, 2010. [Online]. Available: + http://dx.doi.org/10.1093/beheco/arq118

[16] R. Lukeman, Y.-X. Li, and L. Edelstein-Keshet, "Inferring individual rules from collective behavior," *Proceedings of the National Academy of Sciences*, vol. 107, no. 28, pp. 12 576–12 580, 2010. [Online]. Available: http://www.pnas.org/content/107/28/12576.abstract

[17] R. P. Mann, "Bayesian inference for identifying interaction rules in moving animal groups," *PLoS ONE*, vol. 6, no. 8, pp. 1–10, 08 2011.

[18] N. Correll and A. Martinoli, *System Identification of Self-Organizing Robotic Swarms*. Tokyo: Springer Japan, 2006, pp. 31–40.

[19] C. Kunz, "Inferring communication topologies in swarm robotic networks," Master's thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 7 2016.

[20] E. Nickson, "Network inference in swarm robotics," B.S. Thesis, Monash University, Melbourne, Victoria, Australia, November 2016.

[21] A. Maus and J. M. Drange, "All closest neighbors are proper delaunay edges generalized, and its application to parallel algorithms," *Proceedings of Norwegian informatikkonferanse*, pp. 1–12, 2010.

[22] T. Giorgino *et al.*, "Computing and visualizing dynamic time warping alignments in r: the dtw package," *Journal of statistical Software*, vol. 31, no. 7, pp. 1–24, 2009.

[23] M. Andersson, J. Gudmundsson, P. Laube, and T. Wolle, "Reporting leaders and followers among trajectories of moving point objects," *GeoInformatica*, vol. 12, no. 4, pp. 497–528, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10707-007-0037-9

[24] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "Argos: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11721-012-0072-5