Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Obstacle Avoidance of Swarms Using Pinning Control

Kleber M. Cabral [*] **Sidney N. Givigi** [**] **Peter T. Jardine** [*]

[*] *Department of Electrical and Computer Engineering, Royal Military College National, Kingston, ON K7K 7B4 Canada (e-mail: {cabral,peter.jardine}@rmc.ca).*
[**] *School of Computing, Queen's University, Kingston, ON K7L 3N6 Canada (e-mail: sidney.givigi@queensu.ca)*

Abstract: In swarm control tasks, local objectives, on each agent, can interfere with the group's collective objectives. For example, in an environment with obstacles, the motion of the whole group can be affected by local obstacle encounters (happening in a few agents). In this work, we investigate the navigation of swarms in the presence of obstacles. We propose a novel control strategy to avoid obstacles while reducing swarm fragmentation, i.e., limiting the division of the swarm into disconnected groups. We model the swarm as a network where each vehicle is topologically connected with the neighbours that are within the agent's sensing range. We actively monitor the agents' connections in order to identify the necessity of redesigning the network, splitting a larger group into groups with fewer agents. Also, we use a path planning algorithm to provide the trajectory to guide the agents to the final destination. At the end of this paper, we show the results of simulation trials to demonstrate the performance of our control strategy.

Keywords: Pinning control, swarms, obstacle avoidance

1. INTRODUCTION

For some systems, in particular robotic systems, certain advantages automatically arise from using a large number of agents (instead of only one robot). Also, some tasks are impossible to carry out with only one agent. For example, certain assembly tasks. Likewise, using more robots can introduce redundancy in the system (e.g. in area coverage tasks). Also, if needed, operational robots can replace malfunctioning agents (Cortés and Egerstedt, 2017).

Among other applications, for multiple robots, network control methodologies are usually applied in area coverage tasks (Laventall and Cortés, 2009; Notomista and Egerstedt, 2018). Also, the control during rendezvous, alignment, swarming, flocking and formation can be seen in (Olfati-Saber et al., 2007; Wang and Su, 2014; Wang et al., 2016; Cortés and Egerstedt, 2017; Notomista and Egerstedt, 2018). The emergent behaviours of the group arise from the combination of the multiple individual actions.

It is not guaranteed that an emergent behaviour will match with a desired group dynamic. Thus, controlling a multi-agent system adds the capability of guiding the behaviour towards desired states (Liu et al., 2011). To clarify, controlling a network is the process of guiding the states of all nodes to desired values (Nozari et al., 2019). Also, a reliable and efficient network operation can be obtained by controlling the nodes using external inputs (Pasqualetti et al., 2014).

Therefore, obtaining methodologies that enable the control of the multi-agent systems is crucial. In remote operation scenarios (in surveillance tasks, for example), the capability of controlling a large group of robots using as few operators as possible is an attractive problem. We consider an operator as an entity that provides collective objectives to the swarm (and/or monitor swarm's states). Moreover, achieving methodologies that minimize communication between agents (or agents to operator) is also appealing (Liu et al., 2018).

Swarm navigation in the presence of obstacles is one case where collective behaviours may diverge from desired behaviours. Local obstacle encounters (on a few agents) may negatively affect the navigation of the whole group. Obstacle encounters can change the spatial arrangement of a group. As a result, groups can split and connections (communication/sensing) between agents may be broken, causing group fragmentation (Olfati-Saber, 2006).

In this paper, we investigate the coordination of multi-robot systems that navigate in an environment with obstacles. We model the swarm as a network, represented by a graph. We propose an active approach to redesign the network configuration (the logical connection between robots) to reduce the level of swarm fragmentation. Specifically, we study the *group separation* (also called here *group splitting* or *network rupture*) that happens at an obstacle encounter. Instead of considering the separation (fragmentation) of a group at an obstacle encounter, Vrohidis et al. (2017) presented a constraint-based approach for network redesign. The authors show that a possible solution for the problem is the displacement of agents in a straight line (platooning).

We approach the group separation problem by breaking specif network connections, splitting a larger group of agents into smaller groups. We analyze the connectivity between agents to identify the necessity of separating a group. We quantify how close a node is to its neighbours using a connectivity metric. Primarily we want to identify a node that is about to separate from the network, moving away from its neighbours and breaking all of its network connections.

The redesign process is the selection of network connections (edges) that must be broken, yielding to group separation. Here, we actively select the edges to be broken in order to guarantee a minimum number of nodes in the new subgroups (meaning that the agents are a subset of the total agents).

After separation, first, we generate new pins that will control the new sub-networks, and second, we compute the path to guide each sub-network to the final goal. We study here two approaches in order to select new pins. The random selection of a node and the use of the controllability Gramian to select the node that yields the highest controllability. The selection of a control node in a network using the controllability Gramian was presented by (Pasqualetti et al., 2014; Nozari et al., 2019).

Moreover, we use a rapidly-expanding random tree (RRT) algorithm as a path planner. The RRT algorithm is behaving as an operator and provides the collective objective. Also, we apply pinning control strategy to guide the ensemble through the planned trajectory.

The collective behaviour of flocking has been studied in the literature. Olfati-Saber (2006) describes the control laws and analyses the group behaviour for agents tracking a reference and avoiding obstacles. Later on, Wang et al. (2010) proposed the use of pinning controllers applied to swarms, where instead of allowing all nodes to have access to a particular reference, only one node had the information about the desired objective. The term *pinning control* defines the placement of local feedback controllers on a small fraction of the network nodes. Such nodes are called pins or pinned nodes. The nodes that are not directly actuated will be influenced only through their connections with the pins (Zhi-Hong Guan et al., 2010).

The novelties of this work are found in our approach to obstacle avoidance for multi-agent systems. We propose here a metric for evaluating the connectivity of each node. We use the connectivity metric to identify a node that is about to disconnect from the rest of the group (increasing swarm fragmentation). Also, we derive an active methodology for breaking network connections as a method for avoiding obstacles and reduce swarm fragmentation. We base our decision over the group separation on the number of agents in the group and the number of operators available. We consider that each group is connected to one operator, and the number of operators is limited. Therefore, the fragmentation of the swarm should be limited. Moreover, we explore here the dynamic selection of new pins based on two different strategies: i) random selection of a node as pin, or ii) Gramian-based selection. Selecting new pins is critical due to the important role pin agents have. Pins are the only agents that communicate with operators and guide the whole group to desired trajectories. Finally, we use path planning algorithms to act as system operators designing swarm's trajectories.

This paper is organized as follows, Section 2 shows the mathematical description basis of the multi-agent problem. Section 3 shows our control strategy as well as the algorithms proposed. Section 4 shows the main results obtained. Section 5 is the conclusion.

2. PROBLEM FORMULATION

In this section, we present the modelling of swarms of vehicles described as networks. First, we describe here the dynamics of each agent as well as a control law for the multi-agent system. Last, we model the interactions between agents as a dynamical network. This formulation is used later on our proposed control approach (Section 3), and to generate the results in Section 4.

2.1 Agent's modelling and swarm control law

Let us consider a swarm with N agents. Each robot (i), is described as a double integrator,

$$\begin{aligned} \dot{q}_i &= p_i \\ \dot{p}_i &= u_i \end{aligned} \tag{1}$$

where q_i and p_i are respectively the position and velocity of each agent, being $i = 1, 2, ...N$. u_i is the acceleration input signal of the system. q_i, p_i, and $u_i \in \mathbb{R}^M$.

In the literature, the use of double-integrator systems is shown to yield the same results of more complex models (Olfati-Saber, 2006). Also, we use q and p as position and velocity variables (instead of more intuitive letters, such as p and v) to maintain consistency with our references.

Each agent has limited sensing capabilities, with a neighbouring sensor region of radius r. Olfati-Saber (2006) considered that in order to control the position of agents in a flocking, an input signal for each node could be described as the sum of three terms,

$$u_i = u_i^\alpha + u_i^\beta + u_i^\gamma \tag{2}$$

where u_i^α is a network term that determines the interaction with other agents; u_i^β is an obstacle avoidance for collision term; and u_i^γ is a navigational term with the collective objective.

$$\begin{aligned} u_i^\alpha &= c_1^\alpha \sum_{j \in N_i^\alpha} \phi_\alpha(||q_j - q_i||_\sigma)\mathbf{n}_{i,j} + c_2^\alpha \sum_{j \in N_i^\alpha} a_{ij}^q (p_j - p_i) \\ u_i^\beta &= c_1^\beta \sum_{k \in N_i^\beta} \phi_\beta(||\hat{q}_{i,k} - q_i||_\sigma)\hat{\mathbf{n}}_{i,k} + c_2^\beta \sum_{k \in N_i^\beta} b_{i,k}^q (\hat{p}_{i,k} - p_i) \\ u_i^\gamma &= c_1^\gamma \sigma_1(q_r - q_i) + c_2^\gamma (p_r - p_i) \end{aligned} \tag{3}$$

where

- $N_i^\alpha = \{||q_j - q_i|| < r\}$ is the set of nodes connected to node i;
- $N_i^\beta = \{||\hat{q}_{i,k} - q_i|| < r'\}$ is the set of β-agents close to node i. A β-agent is a imaginary agent (with dynamics $\hat{q}_{i,k}, \hat{p}_{i,k}$) at the surface of an obstacle to be avoided. The computation of the dynamics of β-agent is described in (Olfati-Saber, 2006);
- $c_{1,2}^{\alpha,\beta,\gamma}$ are control law gains;

- (q_r, p_r) is the reference or γ-agent;
- $||.||_\sigma$ is the σ-norm. The σ-norm is used instead of the $||.||$ (the Euclidean norm) because it is differentiable everywhere;
- $\phi_\alpha(\cdot)$ is the *attractive/repulsive action function*, used to maintain the distance between agents approximately equal to d;
- $\phi_\beta(\cdot)$ is the *repulsive action function*, used to avoid collision with obstacles;
- a_{ij}^q are adjacency relationships between agents and $b_{i,k}^q$ between agents and obstacles.

We show below how to compute the control law terms in equation (3). Note that, hereafter, we use the Greek letter "χ" to represent the input of the function being described.

First, $\sigma_1(\cdot)$ is

$$\sigma_1(\chi) = (\chi)/\sqrt{1+||\chi||^2} \tag{4}$$

Also, the σ-norm is defined as

$$||\chi||_\sigma = \frac{1}{\epsilon}(\sqrt{1+\epsilon||\chi||^2}-1). \tag{5}$$

with $\epsilon \in (0,1)$.

The vectors $\mathbf{n}_{i,j}$ and $\hat{\mathbf{n}}_{i,k}$ are

$$\begin{aligned} \mathbf{n}_{i,j} &= \frac{q_j - q_i}{\sqrt{1+\epsilon||q_j-q_i||^2}} \\ \hat{\mathbf{n}}_{i,k} &= \frac{\hat{q}_{i,k}-q_i}{\sqrt{1+\epsilon||\hat{q}_{i,k}-q_i||^2}}. \end{aligned} \tag{6}$$

Furthermore, $\phi_\alpha(\cdot)$ and $\phi_\beta(\cdot)$ are

$$\begin{aligned} \phi_\alpha(\chi) &= \rho_\mu(\chi/r_\alpha)\phi(\chi-d_\alpha) \\ \phi(\chi) &= 0.5((a+b)\sigma_1(\chi+c)+(a-b)) \end{aligned} \tag{7}$$

$$\phi_\beta(\chi) = \rho_\mu(\chi/r_\beta)(\sigma_1(\chi-d_\beta)-1) \tag{8}$$

where

- $0 < a \le b$, $c = |a-b|/\sqrt{4ab}$;
- $r_\alpha = ||r||_\sigma$;
- $d_\beta = ||d'||_\sigma$, d' is the allowed distance to an obstacle;
- $r_\beta = ||r'||_\sigma$, r' is the interaction range between agent and obstacle.

The bump function $\rho_\mu(\cdot)$, used in (7) and (8), is a scalar function that smoothly varies between 0 and 1,

$$\rho_\mu(\chi) = \begin{cases} 1, & \chi \in [0,\mu) \\ 0.5(1+cos\left(\pi\dfrac{\chi-\mu}{1-\mu}\right)), & \chi \in [\mu,1] \\ 0, & otherwise \end{cases} \tag{9}$$

with $\mu \in (0,1)$.

Lastly, the adjacency relationships can be written as,

$$\begin{aligned} a_{ij}^q(q_j,q_i) &= \rho_\mu(||q_j-q_i||_\sigma/r_\alpha) \in [0,1], j \neq i \\ b_{i,k}^q(\hat{q}_{i,k},q_i) &= \rho_\mu(||\hat{q}_{i,k}-q_i||_\sigma/d_\beta) \in [0,1]. \end{aligned} \tag{10}$$

Note that a_{ij}^q and $b_{i,k}^q$ depend on the distance between agents (q_j,q_i), distance between agents and obstacles $(\hat{q}_{i,k},q_i)$, the sensing radius r_α, and the desired distance to obstacle d_β.

For a detailed explanation on the meaning of terms described above (bump functions, σ-norm, etc), and a deep analysis of their role on the swarm control task, we kindly refer the reader to (Olfati-Saber, 2006).

2.2 Network description of swarms

The network of agents can be represented as a graph, $G = \{V, E, \mathbf{A}\}$. The node set is defined as $V = \{v_i\}$, and the edge set is $E = \{e_{ij}\}$, $i, j = 1, 2, ..., N$. Two agents are topologically connected if the distance between i and j is smaller than their sensing range r ($\exists e_{ij} \in E\ if\ ||q_j - q_i|| < r$). Moreover, in this paper, we consider that the graph is undirected, thus $e_{ij} = e_{ji}$. The adjacency matrix $\mathbf{A}$ is an $N \times N$ positive semidefinite matrix that describes the network topology and a_{ij} are the elements of $\mathbf{A}$. We consider that $a_{ij} = 1$ if $a_{ij}^q > 0$ (equation (10)), and $a_{ij} = 0$ if $a_{ij}^q = 0$. Also, for undirected graphs, $\mathbf{A}$ is symmetric. Cortes et al. (2005) described this topological connection based on nodal distance as *proximity graphs.*

Wang et al. (2010) proposed that the navigation term, u_i^γ, could be applied only at a fraction of the network agents, called the pin nodes. In the literature, this network control strategy receives the name of *pinning control.* Chen et al. (2007) proved that under certain conditions, as few as only one controller (actuating on a single pin node) is capable of synchronizing the whole network to a target trajectory.

Lastly, let us define h_{ij} as the number of hops (edges) in the shortest path between two nodes. Moreover, we consider the existence of a time delay for a node i to access information of a node j equals to $h_{ij}\Delta t$, Δt being the agent-to-agent communication time.

3. NAVIGATION OF SWARMS WITH ACTIVE MONITORING OF THE NETWORK

In this section, we propose a control strategy to guide the agents from an initial position to a final destination in an environment with obstacles. We present our active approach for splitting the network based on a connectivity metric, the selection of new pins and the control law.

As the agents move during operation, the nodes change their inter-agent distance. As a result, the agents may break and create new connections (edges). The control law, equation (2), does not guarantee that an agent will not break all of its connections, separating itself from the group. For swarm agents in a remote mission, for example, allowing a node to break all of its connections is an undesired behaviour, that raises the necessity of potentially monitoring all agents independently.

In this paper, we consider that the number of operators (monitoring/controlling the swarm) is smaller than the number of agents in the swarm. This can be true, especially in robotic applications. In those, operating a large number of robots is advantageous (e.g., brings redundancy to the system), but communicating with every agent in the swarm may raise feasibility issues (e.g., the number of dedicated communication channels needed).

Therefore, we apply the pinning control strategy due to its characteristic of communicating the reference trajectory only to the pin node. We assume an operator is therefore assigned to each pin. The other nodes in a group are indirectly controlled by their connections with the neighbour agents' (and the pin). Also, we monitor the connections of each agent to guarantee that the number of pins does not exceed the number of available operators.

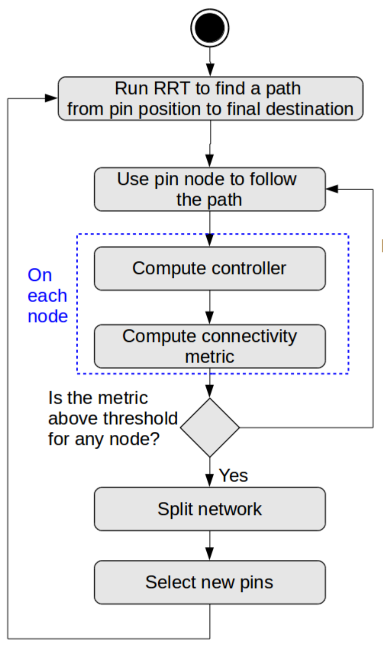

Run RRT to find a path from pin position to final destination
Use pin node to follow the path
On each node
Compute controller
Compute connectivity metric
No
Is the metric above threshold for any node?
Yes
Split network
Select new pins

Fig. 1. Swarm control strategy.

We propose a control strategy based on: a) a path planner that designs a trajectory to guide the robots to the desired position (acting as an operator); b) an active identification of the necessity of separating the network into smaller groups; and c) the use pin nodes to control the swarm following the computed trajectory (reducing the number of nodes that access the reference information).

We show in Fig. 1 our control approach. In the figure, it is possible to see the control strategy described as a sequence of steps. The path planning stage computes a viable trajectory for the swarm. The controller and the connectivity metric are computed at each node, and the decision over the rupture of network connections is based on the connectivity metric. Lastly, after the division of a group, new pins are selected and new paths are computed for each pin.

In the following sections we explain the components of our control strategy: i) the influence of the number of available operators in the system, ii) how to measure the connectivity between agents and how to actively split a group of agents into smaller groups, iii) how to select new pin nodes, and iv) the control law for each agent.

3.1 Operators and RRT algorithm

Let us imagine a remote operation scenario, where each pin is monitored by an operator. This operator represents a remote computer, a person in charge or even a bigger system defining the role of each swarm group takes in a higher-level mission.

In our implementation, the RRT (algorithm 1) computes the trajectory to be followed by the pin node. Lavalle (1998) first introduced the RRT as a viable way of computing a path in a search space. The algorithm works by expanding a tree of possible waypoints from the initial position (the root of the tree) to goal position, avoiding positions that lead to a collision with obstacles. When a leaf of the tree gets closer to the goal, the algorithm selects that branch of waypoints (from root to leaf node) as the path to be followed.

Note that, in our implementation, the RRT algorithm behaves as an operator computing a viable trajectory to the swarm. In the case of a real person as an operator, for example, the trajectory of the agents would be decided at every instant (by the person) and provided to the pin agent.

Algorithm 1 Rapidly-Expanding Random Tree Algorithm

```
1:  procedure RRT($q_{init}, q_{goal}, obstacles$)
2:      Tree.init($q_{init}$), initialize tree at position $q_{init}$
3:      while $q_{goal}$ not reached do
4:          $q_{rand} \leftarrow$ random(), generate a random position
5:          $q_{near} \leftarrow$ find_near($q_{rand}$, G), find nearest vertex
6:          $q_{new} \leftarrow$ new_node($q_{near}$, $q_{rand}$, $\Delta q$), create new node at distance $\Delta q$ of $q_{near}$
7:          if $q_{new}$ does not collide with any obstacle then
8:              Tree.add_vertex($q_{new}$), add node to tree
9:              Tree.add_edge($q_{near}$, $q_{new}$), connect the new node to nearest node
10:         end if
11:     end while
12:     Obtain path from initial position to goal.
13: end procedure
```

Let us consider N^{min} as the minimum number of nodes allowed in a group of agents. We can compute N^{min} by using the number of operators available and the total number of agents in the swarm, $N^{min} = floor(N/\#operators)$. The *floor* function rounds the nearest integer less than or equal to the original number.

3.2 Connectivity Metric and Active Group Separation

The *connectivity metric* of a node (m_i), measures both how many adjacent agents ($e_{ij} \in E$) a node has and how far the agent is from its neighbours. The metric is computed as

$$m_i = \frac{max(||x_j - x_i||)}{d_i + 1}, \; j \in N_i^{\alpha} \tag{11}$$

where $d_i = |N_i^{\alpha}| = \sum_{j=1, j\neq i}^{N} a_{ij}$ is the degree of node i (number of neighbours). The term $max(||x_j - x_i||)$ outputs the distance between i to the farthest node inside its sensing range.

Let us define S as a group of agents in the swarm, with $S \subseteq V$. We assume that there is a path (sequence of edges) in the network graph between any two agents in S.

As a group moves in the environment, their inter-agent distances could change. Therefore their connections (edges in the network topology) could be also dynamically changing. Due to the movement of the agents in space, S can potentially be split into multiple sub-groups, where there is not a path in the graph representation between agents of distinct groups.

Given that N^{min} is the minimum number of nodes allowed in any group, thus $|S| \geq N^{min}$. Furthermore, let us define a *minimal group* as an ensemble with less than $2N^{min}$ agents. That is, there is not a viable way of splitting a minimal group into two new groups without violating

$|S| \geq N^{min}$. Any group with less than $2N^{min}$ agents is considered indivisible.

In our approach, when the connectivity metric of a node (m_i) reaches a threshold, we break specific network connections (removing e_{ij}) in order to split the group (S) into two new groups, for example, S' and S''. Note that, we only remove connections that guarantee that the number of agents in S' and S'' are both bigger than or equal to N^{min}.

3.3 Selection of New Pins

After separation of a bigger group into new smaller groups of agents, we select new pins to act as group leaders and compute their desired trajectory to the goal using algorithm 1. To select new pins, we use two approaches, the random selection of one of the group's agents or the decision based on the controllability Gramian.

Nozari et al. (2019) used the controllability Gramian to select the node that yields to the higher network controllability. The controllability Gramian can be computed as

$$\begin{aligned} \mathbf{C}_{S,T} &:= \left[\mathbf{B}_j \ \mathbf{A}_S\mathbf{B}_j \ ... \ (\mathbf{A}_S)^{T-1}\mathbf{B}_j\right] \\ \mathbf{W}_{S,T} &= \sum_{\tau=0}^{T-1} (\mathbf{A}_S)^{\tau}\mathbf{B}_j\mathbf{B}_K^{\top}((\mathbf{A}_S)^{\top})^{\tau} = \mathbf{C}_{S,T}\mathbf{C}_{S,T}^{\top} \end{aligned} \tag{12}$$

where $\mathbf{A}_S$ is the adjacency matrix of the group S, used as a system dynamics matrix. The $\mathbf{B}_j$ is the column vector with all terms equal to 0, except at the node j (possible node to be used as pin). $(.)^{\top}$ is the transpose operation and T is the control horizon (number of time steps) to control the network.

The trace of the controllability Gramian $(\mathbf{W}_{S,T})$, shows an average controllability value for all the directions in the state space. Thus, choosing the node j that yields the maximum controllability represents selecting the agent that will require less energy to translate the group in space. For more information about this usage of the controllability Gramian, see (Pasqualetti et al., 2014).

3.4 Virtual Nodes and Control Law

In order to maintain cohesion in a minimal group, we introduce here the concept of virtual nodes. A virtual node is computed as the average position of all nodes in a group. Thus, the virtual node is located at the geometric center of the group. The real position and velocity of the virtual node at time t is

$$\begin{aligned} q_{vn}^{r}(t) &= \frac{1}{N^s}\sum_{j\in S} q_j(t) \\ p_{vn}^{r}(t) &= \frac{1}{N^s}\sum_{j\in S} p_j(t). \end{aligned} \tag{13}$$

where S is a minimal group with N^s agents. $N^s < 2N^{min}$.

However, note that an agent accesses the information of another agent in the group delayed by a specific time. Therefore, each node (i) computes the virtual node position and velocity separately as

$$\begin{aligned} q_{vn}^{(i)}(t) &= \frac{1}{N^s}\sum_{j\in S} q_j(t - h_{ij}\Delta t) \\ p_{vn}^{(i)}(t) &= \frac{1}{N^s}\sum_{j\in S} p_j(t - h_{ij}\Delta t). \end{aligned} \tag{14}$$

The virtual node is used to compute the *cohesion control term*, u_i^c, as

$$u_i^c = c_1^{vn}(q_{vn}^{(i)} - q_i) + c_2^{vn}(p_{vn}^{(i)} - p_i) \tag{15}$$

where $c_{1,2}^{vn}$ are the control term gains.

Also, note that in a minimal group, instead of guiding the pin controller to the predefined trajectory, the navigation term is replaced to guide the geometric center of the ensemble (virtual node) to the trajectory. Thus,

$$u_i^{\gamma} = \begin{cases} g_i[c_1^{\gamma}(q_{goal} - q_{vn}^{(i)}) + c_2^{\gamma}(p_{goal} - p_{vn}^{(i)})], \\ \qquad i \in S, S \text{ is a minimal group} \\ g_i[c_1^{\gamma}(q_{goal} - q_i) + c_2^{\gamma}(p_{goal} - p_i)], \\ \qquad i \in S, S \text{ is not a minimal group.} \end{cases} \tag{16}$$

where $g_i = 1$ if i is a pin node, and it is 0 otherwise.

The complete control law signal at each node is then given by

$$u_i = u_i^{\alpha} + u_i^{\beta} + u_i^{\gamma} + u_i^{c} \tag{17}$$

Note that, the u_i^c term is only activated if the group containing i is a minimal group. Thus, maintaining group cohesion while moving through space becomes another objective (along with obstacle avoidance and navigation) of a minimal group.

4. RESULTS

In this section, we show the main results obtained when simulating the proposed swarm control strategy. First, we describe the simulation procedure and parameters used to generate our results. Second, we show the result of the navigation of a swarm of agents using our control strategy (described in Section 3). Finally, we present a comparison between the two pin selection methods proposed in the paper.

For this work, we constrain our analysis to the two-dimensional space, therefore, $q_i, p_i \in \mathbb{R}^2$ with $q_i = \left[q_i^x \ q_i^y\right]^T$ and $p_i = \left[p_i^x \ p_i^y\right]$.

Table 1 shows the parameters used in simulation. Also, in the RRT algorithm, a safe distance to an obstacle (point where a new waypoint is allowed in the path) is r'.

Table 1. Parameters for simulation

Parameters	Value
c_1^{α}, ${}_2^{\alpha}$	2, 2.83
c_1^{β}, c_2^{β}	3, 3.46
c_1^{γ}, c_2^{γ}	1, 2
c_1^{vn}, c_2^{vn}	1, 2
ϵ, μ	0.5, 0.5
d, r	1, 1.3
d', r'	0.6, 0.78
Connectivity Threshold	0.4

Swarm navigation - In the simulation showed in Section 4.1, ten agents perform the navigation in an environment

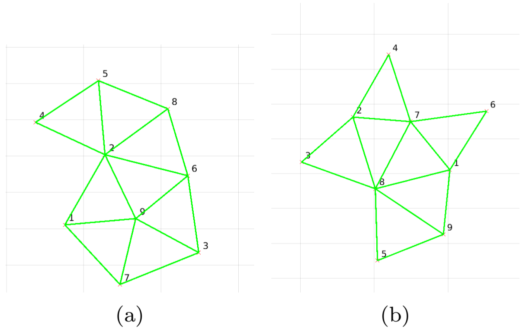


Fig. 2. Example of possible initial configurations, pin selection method comparison.

with obstacles. In the beginning, the agents start in random positions and we establish network connections (e_{ij}) only for every pair of agents closer than r. After that, we selected one agent as the pin node, responsible to follow a desired trajectory and guide the group. Moreover, we considered $\#operators = 3$ (three operators available), resulting in $N^{min} = 3$.

Pin selection method - We executed a simulation of a swarm with nine agents to characterize the error in position tracking using both approaches for pin selection: random agent or Gramian-based selection. To obtain the results showed in Section 4.2, we used the following procedure in simulation:

(1) Start agents in random positions and establish network connections for every pair of agents closer than r. Fig. 2 shows examples of possible initial configurations;
(2) Select a pin node using the Gramian-Based method;
(3) Apply the navigational term, equation (16), on the control law of the pin node to follow a desired trajectory. The trajectory is a sequence of waypoints, 1 meter away from each other. The waypoints define the desired position of the geometric center of the group;
(4) Measure the tracking error, the pin node's control effort, and the distance travelled by each node;
(5) Restart simulation using the same positions obtained in step 1;
(6) Randomly select a node as the pin;
(7) Apply the navigational term on the control law to follow the desired trajectory;
(8) Measure the tracking error, the pin node's control effort, and the distance travelled by each node;

4.1 Swarm navigation

In this section, we show the results of the swarm navigation using our control strategy. Fig. 3 shows snapshots of the autonomous navigation task. It is possible to see the beginning of the task, the moment when the agents break their connections (e.g, Fig. 3c), and the navigation of minimal groups (e.g, Fig. 3h). Furthermore, the figure shows the new pins selected after separation, the creation of virtual nodes, and computation of the agents' paths to the goal. Note that, the trajectory computed is, in fact, a series of sequential waypoints to be reached by a pin node (or a virtual node in a minimal group).

Fig. 4 shows the connectivity metric for each node in the group as they move through the environment with obstacles. In the figure, we highlight the moments where the network topology is actively redesigned (splitting group and selecting new pins). Also, we show the effect of the cohesion term being activated in a minimal group. In Fig. 4 we show two moments where the network is redesigned and groups split. These separation moments can be seen in Fig. 3c and Fig. 3e.

Fig. 5 shows a minimal group during navigation. Note that the group is composed of four agents, the agent number 3 is the pin node, and $N^{min} = 3$. Since $N^{min} < |S| < 2N^{min}$, the group is indivisible (see Section 3.2). Thus, instead of splitting the group, the agents rearrange their positions in order to avoid the obstacles.

4.2 Comparison between pin selection methods

In this section, we show the comparison between the two pin selection methods, Gramian-Based and the Random pin selection (Section 3.3). We performed 100 iterations of 30 seconds of the procedure described in Section 4. Note that, in each iteration of the simulation, we use the same initial configuration (nodal position and network topology) to compare both approaches.

Fig. 6 shows the resulting tracking error (distance between the geometric center of the ensemble and the goal position), during one of the iterations. In the figure, we show only the last ten seconds of simulation to highlight the difference between both methods. I.e, the swarm moved slower to the next waypoint using the randomly selected pin, for the same initial configuration.

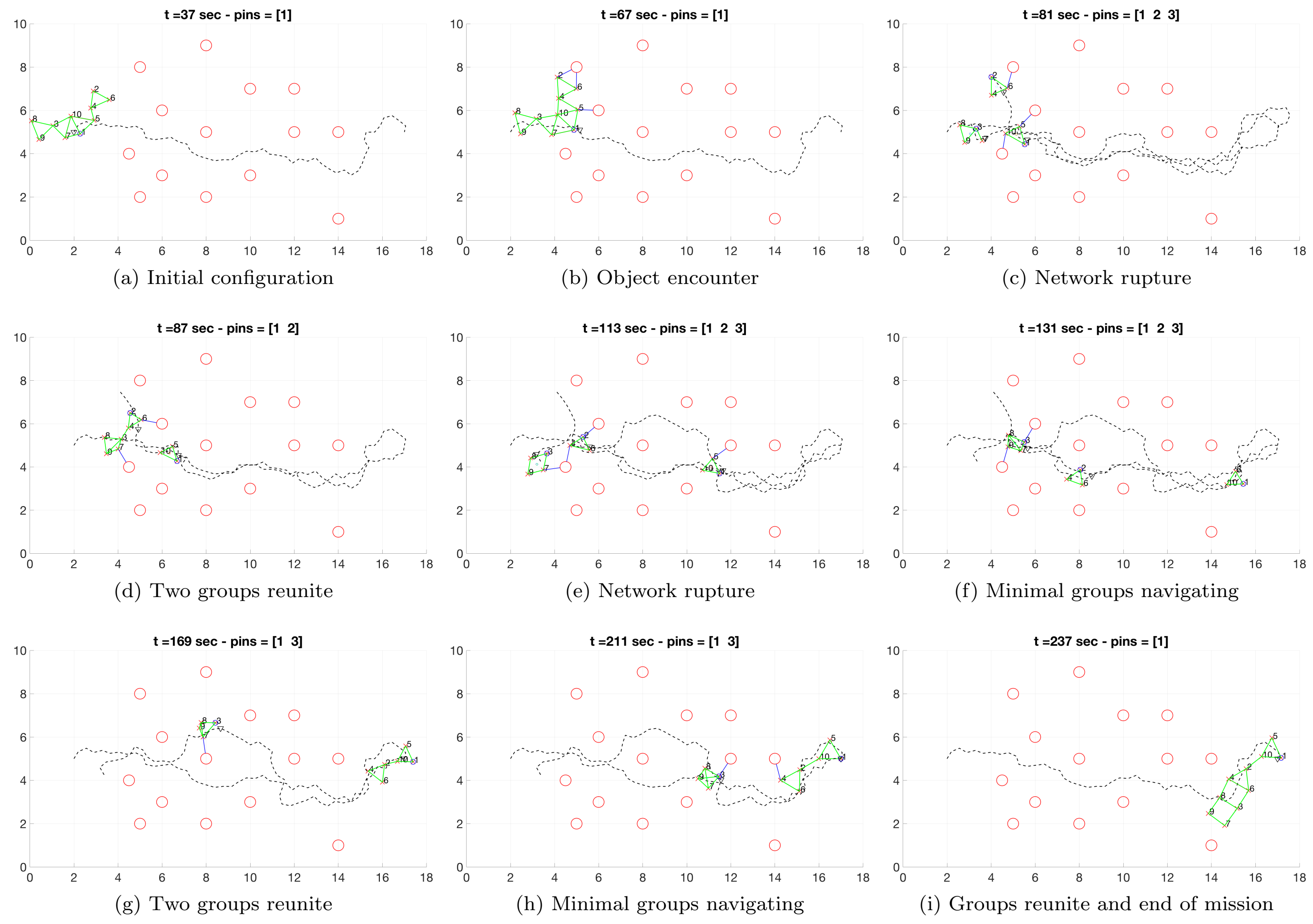
Fig. 7 shows the magnitude of the control effort vector $|u_i|$ for the pin node. In the figure, we can see that the selection of a non-optimal node (randomly), yields to a higher effort to be exerted by the pin node.

Table 2 shows the average RMS value of the Euclidean distance from the swarm geometric center to the target location (tracking error). However, only analyzing the geometric center does not provide a clear idea of the behaviour of the whole swarm during navigation. Therefore, we also measure the distance travelled by each agent. Let us consider q_i^t as the total distance travelled by the node i during operation. Thus, $\bar{q}^t = (1/N)\sum_{i=1}^{N} q_i^t$ is the average distance travelled by all agents in the swarm.

Table 2. Tracking error for different pin selection methods

Pin selection strategy	RMS position error [meters]
Random selection	0.6771
Gramian based	0.6632

Fig. 8 shows a boxplot with the average travelled distance ($\bar{q}^t$) after 100 iterations. We can interpret from the figure that selecting a random node as pin caused all agents in the swarm to move more than using a Gramian-selected pin. The higher travelled distance is the result of the whole group rearranging its positions to accommodate the motion of the pin node. For example, the randomly selected pin node could be located at one end of the network, and its motion may cause the whole group to rotate, or even force network connections to be broken/created.



(a) Initial configuration (b) Object encounter (c) Network rupture (d) Two groups reunite (e) Network rupture (f) Minimal groups navigating (g) Two groups reunite (h) Minimal groups navigating (i) Groups reunite and end of mission

Fig. 3. Autonomous navigation using 10 agents, $\#operators = 3$, and $N^{min} = 3$.

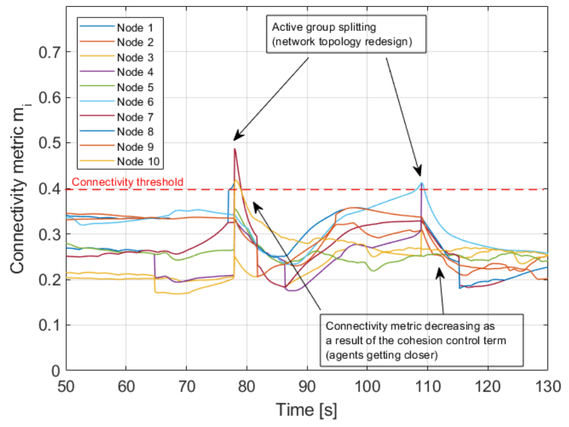


Fig. 4. Connectivity metrics during part of the experiment.

5. CONCLUSION

In this work, we explored the autonomous navigation of swarms in an environment with obstacles. We approached the problem using pinning control (applying the navigational control term in only one node) to guide the swarm to the goal position. We proposed a control strategy that: actively breaks the network connections creating new groups, dynamically defines new pins, and maintains group cohesion while navigating in space. Using simulation trials, we demonstrated our approach effectiveness. Monitoring the agent's connections (distances to neighbours) and actively deciding on splitting a group is shown as a valid approach for navigating an environment with obstacles.

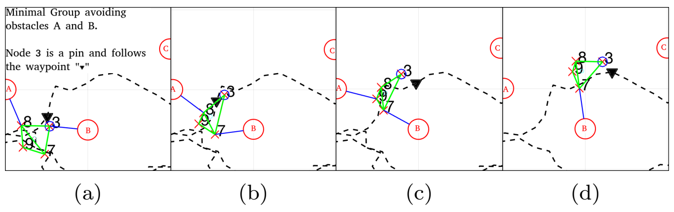

Minimal Group avoiding obstacles A and B.
Node 3 is a pin and follows the waypoint "▾"

Fig. 5. Minimal group avoiding obstacles and maintaining cohesion.

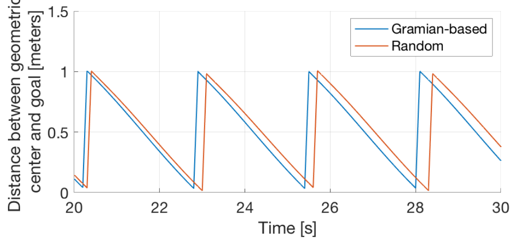


Fig. 6. Tracking error of the geometric center using two different pin selection methods.

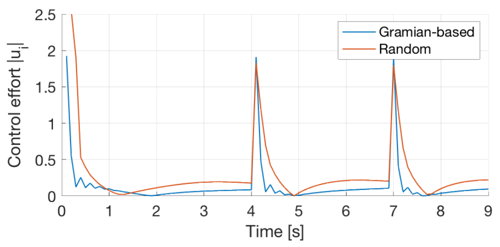


Fig. 7. Control effort of the pin node using two different pin selection methods.

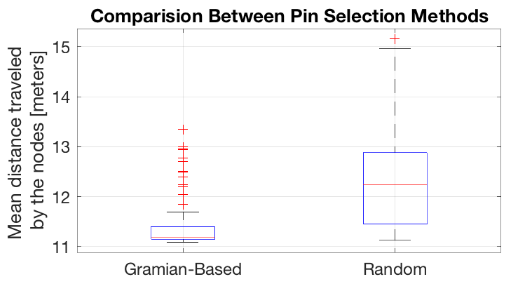


Fig. 8. Result of 100 iterations using the two pin selection methods.

Moreover, the Gramian-based selection of new pins when compared to the random selection showed to be a viable approach for pin selection. The Gramian-based approach resulted in selecting a node that: i) guided the group to its objective with a smaller overall error, ii) required less control energy to control the group, and iii) required less motion (distance travelled) from all agents.

The results presented in this paper can be compared with platooning of agents (Kavathekar and Chen, 2011; Huang and Ren, 1998). Platooning is a viable way of transposing an environment with obstacles, once the vehicles align themselves in a straight line and can avoid obstacles more easily. However, the area covered using platooning is smaller, when compared to the swarm groups presented here. The distributed displacement of the vehicles in a swarm is advantageous in certain applications, such as surveillance. Also, the time delay of the information in the platooning members could be a problem for bigger groups. In our approach, the active rupture of the network and the cohesion control term help to solve such time delay issues.

REFERENCES

Chen, T., Liu, X., and Lu, W. (2007). Pinning complex networks by a single controller. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(6), 1317–1326.

Cortés, J. and Egerstedt, M. (2017). Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, 10(6), 495–503. doi:10.9746/jcmsi.10.495.

Cortes, J., Martinez, S., and Bullo, F. (2005). Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations*, 11(4), 691–719.

Huang, S. and Ren, W. (1998). Longitudinal control with time delay in platooning. *IEE Proceedings-Control Theory and Applications*, 145(2), 211–217.

Kavathekar, P. and Chen, Y. (2011). Vehicle platooning: A brief survey and categorization. In *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 829–845. American Society of Mechanical Engineers.

Lavalle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University.

Laventall, K. and Cortés, J. (2009). Coverage control by multi-robot networks with limited-range anisotropic sensory. *International Journal of Control*, 82(6), 1113–1121.

Liu, Q., He, M., Xu, D., Ding, N., and Wang, Y. (2018). A mechanism for recognizing and suppressing the emergent behavior of uav swarm. *Mathematical Problems in Engineering*, 2018.

Liu, Y.Y., Slotine, J.J., and Barabási, A.L. (2011). Controllability of complex networks. *Nature*, 473(7346), 167–173. doi:10.1038/nature10011.

Notomista, G. and Egerstedt, M. (2018). Constraint-driven coordinated control of multi-robot systems. *arXiv:1811.02465 [cs]*.

Nozari, E., Pasqualetti, F., and Cortés, J. (2019). Heterogeneity of central nodes explains the benefits of time-varying control scheduling in complex dynamical networks. *Journal of Complex Networks*. doi:10.1093/comnet/cnz001.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3), 401–420. doi:10.1109/TAC.2005.864190.

Olfati-Saber, R., Fax, J.A., and Murray, R.M. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1), 215–233. doi:10.1109/JPROC.2006.887293.

Pasqualetti, F., Zampieri, S., and Bullo, F. (2014). Controllability metrics, limitations and algorithms for complex networks. *IEEE Transactions on Control of Network Systems*, 1(1), 40–52. doi:10.1109/TCNS.2014.2310254.

Vrohidis, C., Bechlioulis, C.P., and Kyriakopoulos, K.J. (2017). Safe decentralized and reconfigurable multi-agent control with guaranteed convergence. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 267–272. IEEE.

Wang, X., Zeng, Z., and Cong, Y. (2016). Multi-agent distributed coordination control: developments and directions via graph viewpoint. *Neurocomputing*, 199, 204–218.

Wang, X., Li, X., and Lu, J. (2010). Control and flocking of networked systems via pinning. *IEEE Circuits and Systems Magazine*, 10(3), 83–91.

Wang, X. and Su, H. (2014). Pinning control of complex networked systems: A decade after and beyond. *Annual Reviews in Control*, 38(1), 103–111. doi:10.1016/j.arcontrol.2014.03.008.

Zhi-Hong Guan, Zhi-Wei Liu, Gang Feng, and Yan-Wu Wang (2010). Synchronization of complex dynamical networks with time-varying delays via impulsive distributed control. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(8), 2182–2195. doi:10.1109/TCSI.2009.2037848.