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

From Swarms to Stars: Task Coverage in Robot Swarms with Connectivity Constraints

Jacopo Panerati*, Luca Gianoli†, Carlo Pinciroli‡, Abdo Shabah†, Gabriela Nicolescu*, and Giovanni Beltrame*

***Abstract*— Swarm robotics carries the potential of solving complex tasks using simple devices. To do so, however, one must define distributed control algorithms capable of producing globally coordinated behaviours. We propose a methodology to address the problem of the spatial coverage of multiple tasks with a swarm of robots that must not lose global connectivity. Our methodology comprises two layers: (i) a distributed Robot Navigation Controller (RNC) is responsible for simultaneously guaranteeing connectivity and pursuit of multiple tasks; and (ii) a global Task Scheduling Controller approximates the optimal strategy for the RNC with minimal computational load. Our contributions include: (i) a qualitative analysis of the literature on connectivity assessment, (ii) our proposed methodology, (iii) simulations in a multi-physics environment, (iv) real-life robot experiments, and (v) the experimental validation of connectivity, coverage optimality, and fault-tolerance.**

I. Introduction

Declining costs and thriving popularity are making drones and small robots a recurring sight in our daily lives. This trend, in turn, increases the appeal of multi-robots systems. Swarm robotics carries the potential of solving complex tasks using simple devices—making the whole greater than the sum of its parts. One of its major challenges, however, is the definition of distributed control algorithms capable of producing coordinated behaviours while relying on partial and often noisy information. Swarm robotics exploits control algorithms that make decisions using local information—such as the positions of a robot's direct neighbours. Thus, unlike traditional multi-robot systems, robot swarms execute complex tasks in a distributed fashion, with less reliance on hierarchies or centralization.

Many application scenarios may require a swarm to remain fully connected throughout its entire operational life. The direct consequence, and desirable property, of global connectivity is that a communication path always exists between any two swarm members [1]. However, guaranteeing a swarm's connectivity through a distributed navigation algorithm is not a trivial task. In this work, we propose a hybrid methodology to address the problem of the spatial coverage of multiple tasks by a swarm of robots that never loses global connectivity. Typical applications of this approach would be: (i) the autonomous exploration of extreme and hard-to-reach environments [2]; or (ii) the fast deployment of a communication infrastructure or a GIS in an emergency area [3]. Our results—obtained from mathematical modelling, multi-physics simulations, and laboratory experiments—demonstrate that the modular and distributed nature of our approach can be used to develop autonomous and fault-tolerant mechanisms.

*Polytechnique Montréal, Department of Computer and Software Engineering, {jacopo.panerati, gabriela.nicolescu, giovanni.beltrame}@polymtl.ca

†Worcester Polytechnic Institute, Department of Computer Science, cpinciroli@wpi.edu

‡HumanITas Solutions, Montréal, QC, Canada {luca.gianoli, abdo.shabah}@humanitas.io

II. Literature Review

This section mirrors the two-folded approach of Section III. First, we review previous works tackling the problems of task scheduling and coverage. Then, we present the problem of the distributed assessment of connectivity, including a quantitative review of previous research.

A. On Task Scheduling, Mapping, and Coverage

Literature on the Multi-robot Task Allocation (MRTA) problem is exhaustively analyzed in [4]. Throughout MRTA literature, great relevance is given to temporal and ordering constraints for task execution [5], [6], [7], and [8]. Different objectives function have been proposed, including robot path distance minimization, total duration minimization, and utility maximization [9]. Mixed-Integer Linear Programming (MILP) formulations for different variants of MRTA have been proposed in [8], [5], [6]. In particular, [8], [6] have based their approaches on the centralized heuristic resolution of the proposed MILP formulation. To the best of our knowledge, this work is the first to propose a task scheduling and assignment problem for swarms of robots that jointly includes temporal, geospatial and connectivity constraints and couples it with a distributed lower lever controller.

B. On Swarm Connectivity

Given a swarm of robots that can only communicate with their neighbours, we define (global) *connectivity* as the boolean property that tells us whether a communication path can be established within any two robots. This property is desirable because of its implications (e.g., the ability to implement average consensus [10]) and its distributed assessment and enforcement have been the objective of several previous research works [11], [12].

Connectivity assessment approaches are often based on the formalisms of Spectral Graph Theory (SGT) [13]. These methods describe generic multi-robot systems as graphs $\mathcal{G}=(V,E)$ in which $K=|V|$ robots are represented by nodes

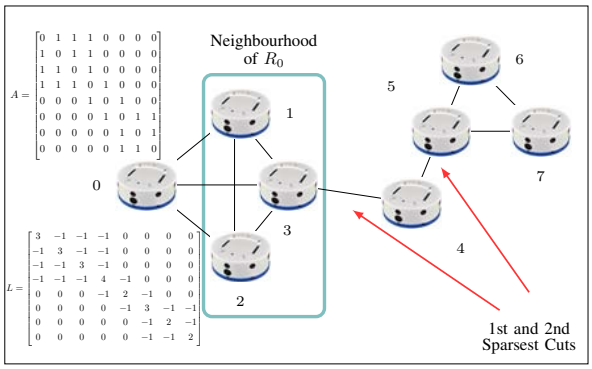


Fig. 1. Robot swarms are often treated as graphs $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ (e.g. *via* Spectral Graph Theory) for networking purposes.

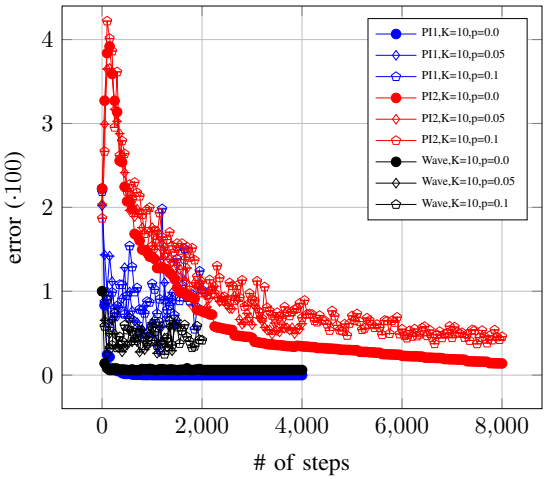


Fig. 2. Comparison of the performance of Spectral Graph Theory methods for the computation of the second eigenvalue of the Laplacian matrix λ_2 under the assumptions of perfect communication packet drop with probability p. PI1 [11], PI2 [14], Wave [17].

and the existing communication links with non-directional arcs $e \in E$. This graphs, in turn, can be represented with the aid of the adjacency matrix A and the Laplacian matrix L (see Fig. 1), tied by the following relationship:

$$L_{K\times K} := D - A = \begin{bmatrix} d_1 & 0 & \dots \\ 0 & d_2 & \dots \\ \dots & & \ddots \end{bmatrix} - \begin{bmatrix} 0 & a_{12} & \dots \\ a_{21} & 0 & \dots \\ \dots & & \ddots \end{bmatrix} \tag{1}$$

where $a_{ij} \in [0, 1]$ expresses whether two robots are neighbours and d_i is the number of neighbours of i. The spectral analysis of the Laplacian matrix reveals insights about the underlying robot network. In particular, the second eigenvalue λ_2 of L represents an upper bound of the sparsest cut of the robot network and the signs of the values in the second eigenvector of L tell us on which side of this cut each robot would lie. SGT can be used as a tool to discover how many link failures a swarm can sustain before losing connectivity. However, it is important to keep in mind that certain desirable connected geometries (e.g., a line of robots) present many sparse cut opportunities.

SGT methods based on power iteration (PI) start from the observation that the PI of L (and matrices directly derived from it) can be computed in a distributed fashion in the form of: $x_k^{i+1} = L_{kk} \cdot x_k - \sum_{j|L_{kj}=-1} x_j^i$. Bertrand and Moonen [11] and Di Lorenzo and Barbarossa [14] suggest different ways to derive a matrix M from L so that each node in a network can compute the second eigenvalue λ_2 of L (and its value of the associated eigenvector). However, these approaches still require non-local information to perform periodical normalization steps and avoid numerical instability. In [11], a beacon node is necessary to broadcast the norm of the second eigenvector; in [14], average consensus is used for the same reason. Other works, such as [15] and [16], consider multiple eigenvalues and topology changes, respectively, but are also limited in their performance by the need to periodically perform consensus.

Rather than beacon nodes or consensus, wave propagation-based methods resort to memory. The method described in [17] can find all the eigenvalues associated to the Laplacian matrix L of a robot network from the Fast Fourier Transform (FFT) of the signal propagated by each robot i using the update: $s_i(t) = 2 \cdot s_i(t-1) - s_i(t-2) + k^2 \sum_{j|L_{ij}=-1} s_j(t-1)$, where k is a constant. The major drawback of this approach, however, is the sensitivity to accurate peak detection in the FFT that make it less suitable for noisy environments.

We used the MATLAB-like scripting language Octave, to compare the number of iterations that are necessary for these methods to converge to a precise estimate of λ_2 in [11], [17], and [14]. The error metric is the percent offset with respect to the actual value of λ_2, i.e., $e = |\dot{\lambda}_2 - \lambda_2|/(10^{-2}\lambda_2)$. Our implementations are available under the MIT license on GitHub[1]. We also evaluated these approaches under the assumption of random packet drop, with probability p, on each of the inter-robot links. The results in Fig. 2 suggest that SGT—despite the appeal of its neat mathematical formulation—might not always be the ideal approach to preserve swarm connectivity for two major reasons: (i) its slow convergence (hundreds of iterations even in small sized $K = 10$ swarms), and (ii) its sensitivity to noise.

III. Methodology

We expose our methodology in two steps: in Subsection III-B, we present a distributed navigation controller and, in Subsection III-C, we introduce a global planning layer.

A. General Overview

Our goal is to lead a swarm of robots to cover multiple spatially distributed tasks without losing global connectivity. For the sake of maintaining swarm connectivity, it is useful to identify two classes of swarm members, i.e., *backbone robots* and *master robots*. The former set's main purpose is to keep the swarm connected. By contrast, master robots can perform special tasks such as (i) influencing backbone robots to move towards specific tasks and (ii) drive the whole swarm

[1] git@github.com:JacopoPan/ar-spectral-graph-theory-comparison.git

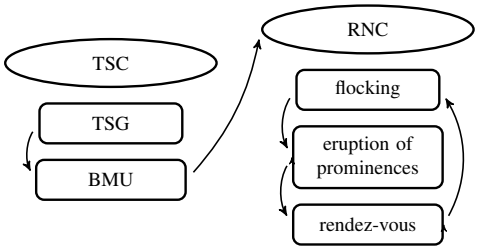


Fig. 3. Flowchart of the overall control architecture.

towards the locations of certain deployment points. Krupke *et al.* [12] proposed a distributed algorithm that drives backbone robots to create trees whose leaves are represented by master robots. Their method resembles ours in its objectives but it is fundamentally different in how it pursues them. In [12], a critical issue is represented by the impossibility to prevent the master robots from excessively stretching the swarm—and thus breaking it—as they drive towards their assigned task locations. To prevent this, we address the problem from a new perspective: pushing robots from well-known deployment points rather than pulling towards unknown tasks. We introduce a swarm task scheduling layer as one of the entity responsible for swarm integrity and we split the swarm control architecture into two parts (Fig. 3).

On one side, the distributed Robot Navigation Controller allows each swarm member to keep a behaviour coherent with its role within the swarm, i.e., that of a master robot or a backbone robot. The Task Scheduling Controller (TSC) is responsible for selecting, at any given time instant, (i) which subset of tasks should be executed by the swarm in a simultaneous fashion, (ii) which robot should be elected master robot, and (iii) how many tasks should be sought from each deployment point.

B. Robot Navigation Controller

The Robot Navigation Controller is the distributed navigation module supporting the following swarm behaviours:

- **Flocking**: the swarm travels toward a new destination while maintaining a compact formation around a single master robot. Once there, the master robot is assigned to the task called Swarm Centroid (SC).
- **Rendez-vous**: the swarm returns to a compact formation around a single master robot. This behaviour is used just before a flocking phase (and just after).
- **Eruption of Prominences**: this behaviour allows the backbone robots to reach distant targets without losing connectivity with the SC. The swarm lays in a fixed formation around the master robot assigned to the SC task. According to the indications received by the TSC layer, the master robot pushes the backbone robots to travel towards different tasks. Backbone robots automatically adjust their position and speed with respect to neighbouring robots to maintain connectivity.

Through this third behaviour, RNC enables the ability to pursue multiple tasks without relying on a global positioning system nor the need to run consensus steps (unlike, e.g., PI-based SGT methods). Connectivity is implicitly but strongly enforced because backbone robots are pushed from the centre by potentials—rather than being pulled from the tasks.

1) Robot Model: All the behaviours implemented by the RNC are based on a robot model that makes very loose assumptions. These are: (i) robots have identical communication and movement capabilities (with a maximum speed v_{max} and constant mass m_r); (ii) they have a limited communication range r; (iii) no information on global positioning (GPS, GLONASS, Galileo, nor BeiDou); and (iv) they can exploit the situated communication model [18]—that is, the ability to assess distance and bearing of close-by robots.

2) Lennard-Jones Potential: The Lennard-Jones potential is a model of inter-atomic interaction that finds frequent use in the context of robotic interaction [19] and it is a low-level component of all of our RNC behaviours. Its force contribution equation is as follows:

$$F_{LJ} = -\epsilon \left(\left(\frac{a \cdot \tau^a}{x^{a+1}} \right)^a - 2 \cdot \left(\frac{b \cdot \tau}{x^{b+1}} \right)^b \right) \tag{2}$$

The parameters ϵ and τ represent the depth of the minimum in the potential and its distance from the origin, respectively.

3) Flocking: The RNC implements a flocking behaviour to allows the swarm to collectively move from one point of deployment (SC) to another. Distributed approaches to flocking are well established in the literature, e.g. [20]:

$$v_i^{t+1} = \sum_{j \in \mathcal{N}} \left(e \cdot x_j^t + f \cdot v_j^t - g \cdot \beth(d_{ij}^t < q) \cdot d_{ij}^t \cdot x_{ij}^t \right) \tag{3}$$

where v_i^{t+1} is the new velocity of robot i at time $t+1$, $\mathcal{N}$ is the set of neighbours of i, x_j^t and v_j^t are the position and velocity of robot j at time t, d_{ij}^t is the distance between i and j, q a threshold, $\beth$ a function that evaluates to 1 if its input condition is met, and e, f, g the cohesion, alignment, separation coefficients, respectively.

4) Rendez-vous: This behaviour is achieved through the broadcast—by the master robot—of two messages, one containing a new, smaller δ parameter to use with (2) and a second message forcing all backbone robots to solely base their control on (2). Backbone robots that receive these messages, further relay the information. The smaller the value of δ, the more compact the resulting swarm.

5) Eruption of Prominences: In [12], the expansion algorithm exploits multiple attraction and repulsion forces to lead backbone robots to build a pseudo-Steiner tree connecting the diverging master robots traveling toward their corresponding task locations. However, we noted that the proposed tree expansion algorithm is very sensitive to several input parameters. This makes the algorithm in [12] unreliable, and thus unfit for use in unknown *a priori* scenarios, such as in disaster relief. In this paper, we propose a new expansion algorithm, the Star Eruption for Connected Swarms (SECS),

that, as the name suggests, leads the backbone robots to form star-like formations that connect several tasks to a central robot, each with a dedicated arm/prominence. Given a subset of tasks to be simultaneously accomplished, each backbone robot is randomly assigned to the arm of a specific task with a probability related to the distance between the location of each task and the swarm centre (known to the master robot). Once assigned to a specific arm, each backbone robot is driven by potential functions and angular correction suggestions (broadcasted by all robots to their neighbours) to find its position within the arm.

SECS provides two major advantages: (i) higher reliability with respect to faults because any robot takes part in providing connectivity with at most one task; and (ii) dependance on only a few input parameters that are not sensitive to the specific features of a scenario.

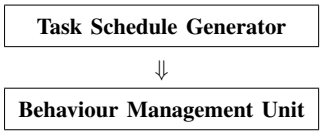
a) From a Distributed to a Shared Coordinate System: The first challenge in creating a coherent prominence (whose absolute direction is only known to the master robot) is the fact that each robot in the swarm possesses its own—and constantly moving—coordinate systems. In order to create collective agreement on the direction of the prominence, all robots j knowing the direction of the prominence in their own coordinate space (initially only the master robot) must propagate two messages: $\angle T_j$ (broadcast) with the local prominence direction and $\angle M_{ji}$ (sent from j to i) with the local direction of i, as seen by j. Then, each robot i can recompute the direction of the prominence in its own coordinate system as: $\angle\ T_i \leftarrow \angle\ M_{ji} - \angle\ O_{ij} + \angle\ T_j$ where $\angle\ O_{ij}$ is the observation of the direction of j in the coordinate system of i.

b) Alignment Maneuver: As soon as a backbone robot learns, from the master or another robot, about the prominence direction $\angle T_i$, it moves towards it. However, this step requires a careful trade-off between: (i) letting all robots pursue the direction of the prominence on their shortest path and (ii) not overcrowding the neighbourhood of the master robot. Thus, we implement the alignment maneuver is by letting each robot *spiraling* (towards $\angle T_i$) around the robot from which it lastly received information about $\angle T_j$ using a randomized parameter (to mitigate collisions).

When a robot finally aligns with the prominence, its control is taken over by the Lennard-Jones potentials of its neighbours. The repulsive forces allow the prominence to erupt towards the desired direction, while the attractive ones prevent the break-down of the swarm. If a robot loses its alignment (e.g., because it was pushed by a neighbouring robot), its control goes back into the spiraling step. A Jupyer Notebook version of the Python code is available under the MIT license on GitHub[2].

C. Task Scheduling Controller

The Task Scheduling Controller is responsible for two main operations: (i) computing the subsets of simultaneous tasks that have to be executed in each specific time slot, and (ii) managing the transition between successive time slots. The TSC can be represented by the two-layer architecture:



The BMU receives from the Task Schedule Generator (TSG) a sequence of task subsets that have to be performed in multiple time slots (one time slot per task subset). The duration of each time slot is low-bounded by the execution time required by the longest task of the corresponding task subset. Each task subset is uniquely defined by its tasks and the deployment point, the point that must be reached in closed formation by the whole swarm before triggering the *eruption of prominences* behaviour. The BMU elaborates the received task sequence to generate the chain of swarm behaviours for the RNC. Let (T_1, T_2) be a sequence of two task subsets, let Δ_1, Δ_2 represent the duration of the longest task for each subset and let Θ_1, Θ_2 be the corresponding deployment points. Assuming the swarm in a compact formation, the swarm behaviour sequence generated by the BMU would be:

1) **Flocking**, to displace the swarm in compact formation from the starting point to deployment point Θ_1.
2) **Eruption of Prominences**, to drive the the backbone robots toward the corresponding task locations for T_1.
3) **Rendez-vous**, after a time Δ_1, to force the whole swarm to regain the original compact formation around the master robot assigned to the SC task placed in Θ_1.
4) Repeat steps 1) to 3) for Θ_2, T_2, Δ_2, and so on.

The TSG takes as input the whole set of tasks requiring completion at a given time and produce a sequence of task subsets for the BMU. The sequence generation must be optimized to reduce the overall execution time and the total distance traveled by the robots, while respecting the hard constraints on the swarm connectivity and reliability. In the following, we show how to formulate the mathematical model representing the optimal task scheduling problem that is solved by the TSG. We discuss a mixed-integer linear programming (MILP) formulation that, with commercial state-of-the-art solvers like IBM CPLEX or GUROBI, can be near-optimally solved in the order of minutes even for large instances with up to 20 robots and 100 tasks.

D. Modelling of the Optimal Task Scheduling Problem

Let us consider a robot swarm operating in a convex region R. Let r^X_{max} (r^Y_{max}) and r^X_{min} (r^Y_{min}) be the real parameters representing the maximum X (Y) coordinate and the minimum X (Y) coordinate that delimit region R, respectively. The set of robots in the swarm is denoted by N. The swarm is initially placed in compact formation around the deployment point of coordinates $\left(\Theta^X_0, \Theta^Y_0\right)$. Being C the set of robot capabilities, e.g., `RGB-camera`, `infrared sensor`, `aerial`, the binary parameter ϕ^c_n is equal to 1 if robot $n \in N$ has capability $c \in C$. We

[2]git@github.com:JacopoPan/ar-prominences-in-the-ideal-world.git

also assume that all the robots are equipped with the same communication technology, that guarantees robot-to-robot communication within a maximum communication range of value Γ. Capability parameters ϕ_n^c are elaborated to define a set K of configuration classes, where each class contains all the robots with the same set of capabilities.

The robot swarm is asked to accomplish the set of tasks T. A task $t \in T$ is characterized by the location coordinates (σ_t^X and σ_t^Y), the list of required robot capabilities, the number of demanded robots (β_t), and the expected task duration (δ_t). The binary parameter υ_t^c is equal to 1 if task $t \in T$ requires a robot with the capability $c \in C$. This parameter is exploited to define the new binary parameter τ_{tk}, equal to 1 if a robot of configuration class k is able to perform task t.

A task priority scheme may also be enforced through binary parameters μ_{t2}^{t1}, which are equal to 1 if task $t1 \in T$ has to be executed before task $t2 \in T \setminus \{t1\}$. We denote with S the ordered set of consecutive time-slots. Note that the integer parameter a_s is equal to the position of time slot s in the ordered set S. To guarantee reliability to robot failures or temporary communication impairments, it can be desirable to guarantee the existence of Ω robot-disjoint communication paths between each task and the central robot.

1) Total Completion Time Minimization: We exploit the aforementioned notation to formalize a MILP formulation for the Total Completion Time Minimization (TCTM) that guarantees SECS-based swarm connectivity throughout the entire duration of the scenario. Let z^s be the non-negative real variables representing the duration of time-slot $s \in S$; note that z^s is kept to 0 if a time-slot contains no task. Furthermore, let x_t^s be the binary variables equal to 1 if task $t \in T$ is scheduled for execution in time-slot $s \in S$. The time minimization objective function can be expressed as:

$$\Lambda = \min \left\{ \sum_{s \in S} z^s \right\} \tag{4}$$

The total duration z^s of time-slot $s \in S$ is bounded from below by the expected completion time of the longest task scheduled for execution during that time-slot. This relation is expressed by:

$$z^s \geq \delta_t x_t^s, \qquad \forall s \in S,\, t \in T \tag{5}$$

Each task must be scheduled in exactly one time-slot:

$$\sum_{s \in S} x_t^s = 1, \qquad \forall t \in T \tag{6}$$

We introduce the non-negative real variables y_t^{Xs} and y_t^{Ys} to account for the X-axis and Y-axis distances between a task $t \in T$ and the swarm deployment point chosen for time-slot $s \in S$. A task can be executed during a specific time-slot only if enough backbone robots are available to build an arm that connects the task and the master robot assigned to the SC task. The distance between the backbone robots in an arm must be smaller than the communication range Γ, and Ω robot-disjoint paths should exist between the edge and the centre. To determine the maximum distance that can be covered by an arm, we introduce the non-negative integer variables w_t^s representing the number of robots that will form the arm of task $t \in T$ during time-slot $s \in S$. Ideally, the distance constraint should be expressed by a classic circle equation such as:

$$\left(y_t^{Xs}\right)^2 + \left(y_t^{Ys}\right)^2 \leq \left(\Gamma w_t^s\right)^2$$

However, such non-linearity should be avoided to not dramatically increase the problem complexity. To this purpose, instead of bounding distance variables y with a circle of radius Γw_t^s, we approximate the positive quadrant of the circle through three linear pieces, i.e., the vertical line passing at $(\Gamma w_t^s, 0)$, the horizontal line passing at $(0, \Gamma w_t^s)$, and the diagonal line passing at $\left(\sqrt{\Gamma w_t^s/2}, \sqrt{\Gamma w_t^s/2}\right)$ with first derivative equal to -1. The three contributions are modelled through the three groups of constraints below:

$$y_t^{Xs} + y_t^{Ys} \leq \frac{\sqrt{2}\Gamma}{\Omega}\left(w_t^s + \Omega\right) + M\left(1 - x_t^s\right), \forall s \in S,\, t \in T \tag{7}$$

$$y_t^{Xs} \leq \frac{\Gamma}{\Omega}\left(w_t^s + \Omega\right) + M\left(1 - x_t^s\right), \forall s \in S,\, t \in T \tag{8}$$

$$y_t^{Ys} \leq \frac{\Gamma}{\Omega}\left(w_t^s + \Omega\right) + M\left(1 - x_t^s\right), \forall s \in S,\, t \in T \tag{9}$$

M makes the constraints useless when the task is not executed in the considered time slot. The total number of robots required to execute a subset of tasks scheduled during the same time slot while keeping the connectivity, must not exceed the total number of swarm member $|N|$:

$$\sum_{t \in T} \left(w_t^s + \beta_t x_t^s\right) \leq |N| - 1, \qquad \forall s \in S \tag{10}$$

The following valid inequalities, stating that backbone robots are assigned only to arms of active tasks, can be included to reduce the solution space and speed up the solution computation:

$$w_t^s \leq \left(|N| - 1\right) x_t^s, \qquad \forall s \in S,\, t \in T \tag{11}$$

Another group of constraints can be added to force the solution to use the first available time slot, therefore restricting the solution space:

$$\sum_{t \in T} x_t^s \leq |T| \sum_{t \in T} x_t^{s-1}, \qquad \forall s \in S \tag{12}$$

Finally, a last group of constraints is included to respect the priorities defined by parameters μ:

$$\sum_{s1 \in S} a^{s1} \mu_{t2}^{t1} x_{t1}^{s1} \leq \sum_{s2 \in S} a^{s2} x_{t2}^{s2} - 1, \quad \forall t1, t2 \in T \tag{13}$$

2) Total Dislocation Space Minimization: Once the minimum completion time is minimized by the TCTM formulation, we propose to address a second optimization problem to minimize the overall robot dislocation space while guaranteeing the optimal completion time previously computed. Let $\bar{S}$ be the time slot ordered set that does not include the first

time slot and let S_0 be the time slot set containing only the first slot. Furthermore, let $\bar{g}^{Xs}$, $\bar{g}^{Ys}$ be the non-negative real variables representing the X and Y distances, respectively, between the deployment point of time slot $s \in S$ and that of time slot $s-1$; The objective function of the Total Dislocation Space Minimization (TDSM) problem is expressed by:

$$\min \left\{ |N| \sum_{s \in \bar{S}} \left(\bar{g}^{Xs} + \bar{g}^{Ys} \right) + \sum_{s \in S, t \in T} \left(\frac{\Gamma}{\Omega} w_t^s \right) \right\} \qquad (14)$$

where, the first term is the overall inter-deployment point dislocation distance, and the second term approximates the length of each active arm. For both TCTM and TDSM, a number of additional constraints, as well as the variables domains, are omitted here due to space limitations.

IV. Experimental Set-up

To validate the RNC and TSG presented in the previous section, we performed the following experiments.

A. Robot Navigation Controller

The SECS algorithm exploited by the RNC was first implemented in an idealized scenario—in which we assumed no robot collisions nor communication interferences—using Python and Jupyter Notebook[3]. After the initial results validated SECS, we moved to its implementation in a more realistic scenario with the aid of the multi-physics robot simulator ARGoS [21]. ARGoS can efficiently simulate large-scale swarms of robots of any kind and it model complex real-life interactions, including collisions, inertia, robots obstructing the sight, movement, and communication of other robots, etc. ARGoS supports robot controllers written in C++, however, for our second implementation of SECS we chose to use Buzz, an internally developed and swarm-specific programming language [22]. Using ARGoS and Buzz, we performed simulations in which the number tasks T_a was varied between 1, 3, and 5. For the number of robots N_a, we used swarms of size 3, 5, 10, 15, 20 and 25 (backbone robots, not including the master robot). The arena we used for the simulations was $D_a = 30$ meters in length and width. The maximum linear speed of each robot ($\sim$ 10cm in diameter) was limited to 15cm/s. The communication range between robots r_a was set at 3 meters. We observe that, despite $D_p \neq D_a$ and $r_p \neq r_a$, the ratios D_p/r_p and D_a/r_a are comparable, meaning that the robots have similar room to move in the Python and ARGoS simulations. With regard to the Lennard-Jones potential parameters, we used $\epsilon = 10^6$, $\delta \in [90.0, 120.0, 240.0, 275.0]$, and exponents of 4 and 2. In our discussion, we explore how the choice of the δ parameter affected the ability of SECS to create prominences of different density (and how this echoed on their propagation time and reliability/link redundancy). All these implementations are available under the MIT license on GitHub[4]. Beyond the simulations, we also evaluated the RNC and SECS in the lab using seven Khepera IV robots (see our video attachment `ICRA2018.mp4`).

[3] git@github.com:JacopoPan/ar-prominences-in-the-ideal-world.git

[4] git@github.com:JacopoPan/ar-argos-buzz-simulations.git

B. Task Scheduling Controller

Both TCTM and TDSM were tested over 720 random instances of the problem. The experiments were carried out on machines equipped with an Intel® Core™ i7-3770 and 32 GB of RAM. We relied on AMPL as modelling language, while we used CPLEX 12.7 with `threads = 8` and `mipemphasis = 1` to solve the MILP formulations. For each instance, we considered a time-limit of 3 hours and solved, sequentially, the TCTM and the TDSM problems. For TDSM we used the maximum duration Λ returned by TCTM. All instances were characterized by a square arena of side L, where: $L = r^X_{max} - r^X_{min} = r^Y_{max} - r^Y_{min}$.

L was determined as in [23], according to the communication density value D and the robot cardinality $|N|$:

$$L = \sqrt{\frac{N\pi\Gamma^2}{D}} \qquad (15)$$

In each instance, we considered a capability set C of cardinality 2. We used p_{Rob} and p_{Tsk} in $[0, 1]$ to determine the probability for a robot of having a capability and that for a task of demanding a capability (same probability for each capability of the set). Furthermore, let β_{max} and δ_{max} be the maximum number of robots that a task may require and the maximum allowed task duration. During the generation, β and δ values were chosen according to a uniform distribution within the integer set delimited by 1 and the corresponding maximum value. The location of each task $\left(\sigma^X_t, \sigma^Y_t\right)$, as well as the starting deployment point $\left(\Theta^X_0, \Theta^Y_0\right)$ were chosen uniformly within the square arena. We considered no priority among the tasks (μ parameters all equal to 0). Both communication range Γ and reliability value Ω were fixed to 1.

V. Experimental Results and Discussion

A. Robot Navigation Controller

The results of the ARGoS experiments, such as the one presented in Fig. 4 ($N = 10$, $T = 1$), show that the algorithm Buzz implementation succeeds in its intent and it is capable of overcoming issues such as communication interference—the loss of a few messages does not prevent the eventual insurgence of the desired behaviour. Fig. 4(a) presents and example scenario in which $\delta = 90$ as a Lennard-Jones parameter makes the robot stick close together and create thicker prominences. These kind of prominences reach less further but are more reliable to robot failures (higher link redundancy). Fig. 4(b) displays the same experiment but with a δ parameter if 240. This suffices for the creation of much thinner prominences that can achieve tasks in more remote locations. In Fig. 5, we provide the results of an experiment that demonstrates how SECS can drive a mid-sized swarm ($N = 16$) towards multiple tasks ($T = 3$) that are located at the same distance from the swarm centroid and have angular separation of 120° from one another. The value of δ used in

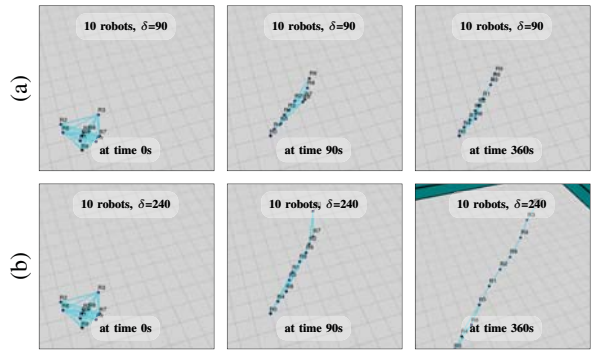


Fig. 4. The RNC (implemented with Buzz programming language and tested in the ARGoS simulation environment) produces eruption of prominences with the same number of robots different lengths in relation to the value of δ—90 in (a), 240 in (b)—used to parameterized the neighbour potentials.

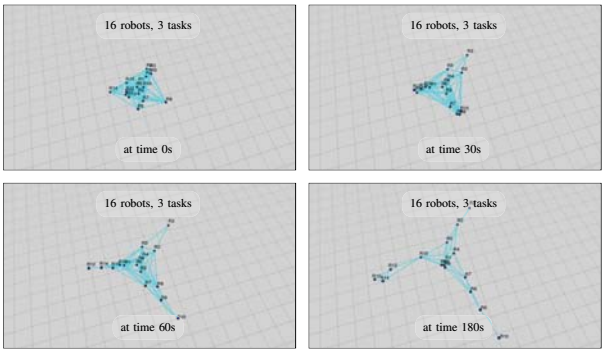


Fig. 5. The RNC (implemented with Buzz programming language and tested in the ARGoS simulation environment) drives a swarm of 16 robots towards three different task directions from the swarm centroid.

this test is 240 and the time to reach fully deployment is in the order of a few minutes ($\sim 180''$).

Table I summarizes the averages of the results of the experiments we conducted with Lennard-Jones potentials parametrized by $\delta = 120$ $\delta = 240$. The size of the swarm was varied from 3 to 20 robots. As we would expect, there is visible correlation both (i) between the number of robots in a prominence and the length of the prominence and (ii) between the value of δ and the length of the prominence (i.e., linear trends down the columns and between corresponding entries in the top and bottom part of the table). With regard to convergence and latency, we observe that SECS still scales well: more robots do not slow-down (but speed-up) the creation of short prominences, in the order of $\sim 10^1$ seconds. The full stretch of a prominence with $\sim 10^1$ robots requires $\sim 10^2$ seconds. With regard to fault tolerance, we report the densities of the prominences (in robots/meter) to show that even the longest prominences (> 20 meters have density $> 0.6\bar{6}$ that, with $r = 3$ means, on average, a double path on every communication link.

Finally, we implemented and tested some of crucial steps in SECS, e.g., the exchange of coordinate systems and the alignment maneuver, in a real-world robot simulation using up to seven Khepera IV robots—as shown in the video attachment of this article `ICRA2018.mp4`.

TABLE I

AVERAGE ARGOS/BUZZ PROMINENCE ERUPTION TIMES

δ	Number of Robots	Latency (s) and Density (robot/m) $\rightarrow 2\sqrt{2}$m	$\rightarrow 3\sqrt{2}$m	$\rightarrow 10\sqrt{2}$m	$\rightarrow 15\sqrt{2}$m
120	3	n/a	n/a	n/a	[$<$ 1.89m]
	5	81.1, 1.77	n/a	n/a	[$<$ 3.39m]
	10	36.3, 3.54	69.8, 2.36	n/a	[$<$ 7.31m]
	15	37.9, 5.30	56.4, 3.54	n/a	[$<$ 12.02m]
	20	50.9, 7.07	63.2, 4.71	345.0, 1.41	[$<$ 18.38m]
	Number of Robots	Latency (s) and Density (robot/m) $\rightarrow 3\sqrt{2}$m	$\rightarrow 5\sqrt{2}$m	$\rightarrow 10\sqrt{2}$m	$\rightarrow 15\sqrt{2}$m
240	3	69.8, 0.71	n/a	n/a	[$<$ 4.95m]
	5	63.7, 1.18	195.1, 0.71	n/a	[$<$ 7.78m]
	10	57.6, 2.36	100.1, 1.41	n/a	[$<$ 11.31m]
	15	41.9, 3.54	74.2, 2.12	327.4, 1.06	449.8, 0.71
	20	43.3, 4.71	86.1, 2.83	237.2, 1.41	322.6, 0.94

B. Task Scheduling Controller

We observe that the TCTM and TDSM (for which an example solution is given in Fig. 6) performance is correlated with the instance features. The total duration values reveal, as expected, that the higher the robot density, the lower the optimal execution time. The computation times are reported in Table II. The two main insights are: (i) the instances get easier to be solved as the communication density D grows, (ii) the TDSM formulation is computationally more expensive than the TCTM formulation. The complexity related to the location of each deployment point and to the arm lengths is the factor that makes the TDSM formulation more computational expensive. However, this additional computation is often well allocated, as shown in Fig. 7, TDSM optimal solutions drastically decrease the dislocation spaces that all the element of the swarm must cover between successive time slots. The same improvement is not observed in the case the arm costs (second term of Objective function 14) reported in Fig. 8.

It strongly stands out that heterogeneous task durations δ (class 5) make both TCTM and TDSM more computationally expensive, which results in the time-limit being reached in most of the instances (the returned solutions have a gap from the best lower bound found at the moment of the time-limit expiration); heterogeneous task durations make the implicit bin-packing problem more complex (it is better to match together in the same time-slot all the longest tasks) and make the Linear Programming (LP) relaxation used to search for integer solutions and improve the lower bound significantly less efficient.

TABLE II

COMPUTATIONAL TIMES

	TCTM					
ID	0.005	0.01	0.05	0.1	0.2	0.5
1	908,1	206,4	15,3	7,0	4,9	0,8
2	210,7	62,8	3,8	3,6	2,8	0,7
3	1488,4	479,6	13,6	12,2	1,7	1,9
4	1362,1	202,2	27,9	44,8	2,1	3,3
5	8038,6	9626,5	6350,4	1006,1	134,1	9,0
	TDSM					
ID	0.005	0.01	0.05	0.1	0.2	0.5
1	2455,4	1657,7	811,8	488,0	2254,3	44,3
2	1036,9	1111,5	530,8	174,2	2332,1	50,5
3	2817,4	1998,0	524,9	1338,8	772,1	1049,3
4	4890,9	2200,1	781,1	2149,4	1249,0	1453,8
5	10800,0	10800,0	10800,0	10094,7	5719,1	31,6

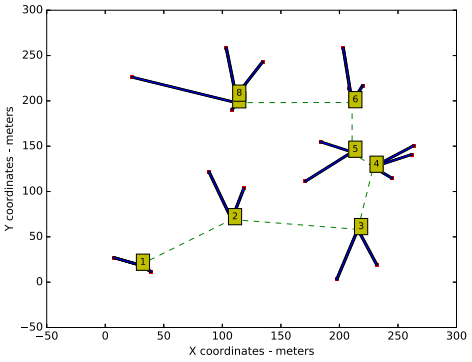


Fig. 6. Example of TSDM solution (with 100 robots and 20 tasks) that uses deployment points (SCs) associated to two to four tasks.

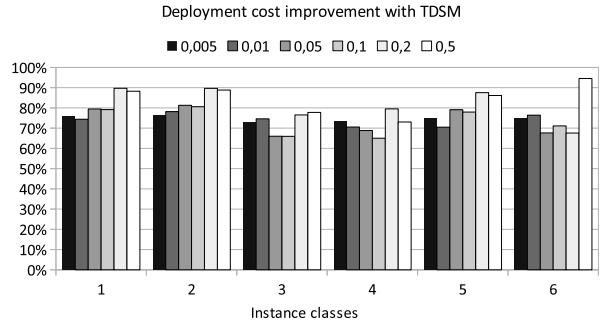


Fig. 7. Analysis of the improvement achieved by TDSM w.r.t. to TCTM in terms of dislocation costs, i.e., $\sum_{s\in\bar{S}}\left(\bar{g}^{Xs}+\bar{g}^{Ys}\right)$.

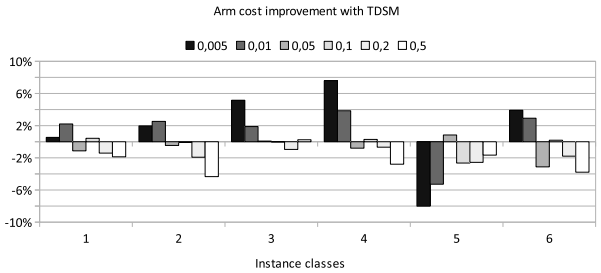


Fig. 8. Analysis of the improvement achieved by TDSM w.r.t. to TCTM in terms of arm costs, i.e., $\sum_{s\in S,t\in T}\left(w_t^s\Gamma\ \Omega\right)$.

VI. Conclusions and Future Work

In this work, we proposed a hybrid methodology to address the problem of the spatial coverage of multiple tasks using a swarm of robots that preserve their global connectivity. Our approach has beed implemented *via* two layers: the Robot Navigation Controller that guarantees the connectivity of the swarm; and a global planner (TSC) that approximates the best strategy for the RNC for two optimization metrics—TCTM and TDSM. We tested the RNC using a multi-physics environment and through laboratory experiments. The TSC experimental results showed that it is possible to find near-optimal coordination strategies for the RNC with affordable computational times. In the future developments of this research, we plan on extending the RNC with additional complex behaviours (e.g., the ability to build slime mold and fractal-like formations).

References

[1] A. R. Mosteo, L. Montano, and M. G. Lagoudakis, "Multi-robot routing under limited communication range," in *2008 ICRA Proceedings*, May 2008, pp. 1531–1536.

[2] D. S. Katz and R. R. Some, "Nasa advances robotic space exploration," *Computer*, vol. 36, no. 1, pp. 52–61, Jan 2003.

[3] L. Apvrille, T. Tanzi, and J. L. Dugelay, "Autonomous drones for assisting rescue services within the context of natural disasters," in *2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*, Aug 2014, pp. 1–4.

[4] E. Nunes, M. Manner, H. Mitiche, and M.Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, pp. 1–45, oct 2016.

[5] L. Luo, N. Chakraborty, and K. Sycara, "Multi-robot assignment algorithm for tasks with set precedence constraints," in *2011 ICRA Proceedings*, May 2011, pp. 2526–2533.

[6] G. Korsah, B. Kannan, B. Browning, A. Stentz, and M. Dias, "xBots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies," in *2012 ICRA Proceedings*, May 2012, pp. 115–122.

[7] E. Nunes and M. Gini, "Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints," *AAAI*, pp. 2110–2116, 2015.

[8] M. Gombolay, R. Wilcox, and J. Shah, "Fast Scheduling of Multi-Robot Teams with Temporospatial Constraints," *Robotics: Science and Systems*, 2013.

[9] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, June 2005.

[10] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33 – 46, 2007.

[11] A. Bertrand and M. Moonen, "Distributed computation of the fiedler vector with application to topology inference in ad hoc networks," *Signal Processing*, vol. 93, no. 5, pp. 1106 – 1117, 2013.

[12] D. Krupke, M. Ernestus, M. Hemmer, and S. P. Fekete, "Distributed cohesive control for robot swarms: Maintaining good connectivity in the presence of exterior forces," in *2015 IEEE/RSJ IROS Proceedings*, Sep. 2015, pp. 413–420.

[13] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, Sept 2011.

[14] P. D. Lorenzo and S. Barbarossa, "Distributed estimation and control of algebraic connectivity over random graphs," *IEEE Transactions on Signal Processing*, vol. 62, no. 21, pp. 5615–5628, Nov. 2014.

[15] M. Zareh, L. Sabattini, and C. Secchi, "Enforcing biconnectivity in multi-robot systems," 2016. [Online]. Available: http://arxiv.org/abs/1608.02286

[16] H. Gruhn and P. Persson, "Towards a robust algorithm for distributed monitoring of network topology changes," in *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, Jun. 2014, pp. 1–7.

[17] T. Sahai, A. Speranzon, and A. Banaszuk, "Hearing the clusters of a graph: A distributed algorithm," *Automatica*, vol. 48, no. 1, pp. 15–24, Jan. 2012.

[18] K. Støy, "Using situated communication in distributed autonomous mobile robotics," in *Proceedings of the Seventh Scandinavian Conference on Artificial Intelligence*, ser. SCAI '01. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2001, pp. 44–52.

[19] 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.

[20] C. Reynolds, "Steering behaviors for autonomous characters," 1999.

[21] 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.

[22] C. Pinciroli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *Proceedings of the IEEE/RSJ IROS 2016*. Los Alamitos, CA: IEEE Computer Society Press, Oct. 2016.

[23] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*. ICST, 2016, pp. 287–294.