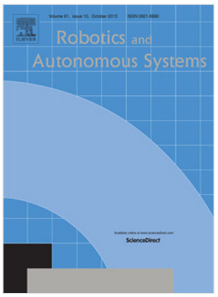
Journal Pre-proof



Self-optimization of resilient topologies for fallible multi-robots

Marco Minelli, Jacopo Panerati, Marcel Kaufmann, Cinara Ghedini, Giovanni Beltrame, Lorenzo Sabattini

PII: S0921-8890(19)30190-3
DOI: https://doi.org/10.1016/j.robot.2019.103384
Reference: ROBOT 103384

To appear in: *Robotics and Autonomous Systems*

Please cite this article as: M. Minelli, J. Panerati, M. Kaufmann et al., Self-optimization of resilient topologies for fallible multi-robots, *Robotics and Autonomous Systems* (2019), doi: https://doi.org/10.1016/j.robot.2019.103384.

Self-optimization of Resilient Topologies for Fallible Multi-robots

Marco Minelli[a], Jacopo Panerati[b,*], Marcel Kaufmann[b], Cinara Ghedini[c], Giovanni Beltrame[b], Lorenzo Sabattini[a]

[a] *Department of Sciences and Methods for Engineering, Università degli Studi di Modena e Reggio Emilia, Reggio Emilia, Italy*
[b] *Department of Software and Computer Engineering, Polytechnique Montréal, Québec, Canada*
[c] *Departamento de Computação Científica, Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil*

Abstract

Effective exchange of information in multi-robot systems is one of the grand challenges of today's robotics. Here, we address the problem of simultaneously maximizing the (i) resilience to faults and (ii) area coverage of dynamic multi-robot topologies. We want to avoid the onset of single points of failure, i.e., situations in which the failure of a single robot causes the loss of connectivity in the overall network. Our methodology is based on (i) a three-fold control law and (ii) a distributed online optimization strategy that computes the optimal choice of control parameters for each robot. By doing so, connectivity is not only preserved, but also made resilient to failures as the network topology evolves. To assess the effectiveness of our approach, we ran experiments with a team of eight two-wheeled robots and we evaluated it against the injection of two separate classes of faults: communication and hardware failures. Results show that the proposed approach continues to perform as intended, even in the presence of these hazards.

Keywords: fault-tolerance, resilience, multi-robot systems, connectivity, graph theory, control, online optimization, robotic hardware

2010 MSC: 00-01, 99-00

*Corresponding author: jacopo.panerati@polymtl.ca

1. Introduction

In this paper, we consider the problem of achieving resilience in a system composed by multiple robots using a wireless network to exchange data and coordinate towards a common goal. Resilience was defined in [1] as a property "about systems that can bend without breaking. Resilient systems are self-aware and self-regulating, and can recover from large-scale disruptions". In this paper we consider the effect of single robots' failures and unreliable communication on the overall performance of the multi-robot system: resilience is then represented by how gracefully the performance of the overall system decreases, in the presence of such failures.

A key ingredient for achieving resilience to failures is redundancy: the presence of multiple entities that can achieve a task leads to the possibility of success, even with the failure of a limited number of such entities. This is a trait of swarms and multi-robot systems, where the overall capability of the system is achieved as the combination of the capabilities of single robots. However, having a large number of robots *per se* does not imply redundancy (and thus resilience). In fact, situations may exist in which even a single robot has a critical role, and its failure renders the completion of a task impossible.

For instance, in groups of heterogeneous robots each robot has different capabilities (with regard to sensing, mobility, actuation, etc.). If all the capabilities are required, then task completion can become impossible as soon as one of the robots stops working. This issue can be effectively mitigated by replicating capabilities across the swarm [2].

Nevertheless, even in the context of completely homogeneous groups, critical robots may still exist. To cooperate and achieve shared objectives, robots need to exchange information. This is possible only when the graph that represents the communication topology among the robots is connected. Connectedness is particularly critical when considering groups of mobile robots with limited-range communication capabilities, since the topology of the network changes as the robots move. Hence, constraints need to be imposed on the robots' motion

in such a way that connectivity is preserved.

This problem has been extensively studied in literature, and several procedures for connectivity preservation have been proposed [3, 4, 5, 6, 7, 8, 9]. These strategies typically start from the assumption that the communication graph is initially connected and they guarantee the preservation of this property as the system evolves. However, those strategies generally do not consider robot failures. As a consequence, pathological situations often exist in which, based on the current topological configuration of the network, failure of a single node leads to the disconnection of the network, and the creation of two (or more) isolated sub-networks. The presence of such critical nodes completely defeats the inherent redundancy of homogeneous multi-robot systems.

In [10], the authors propose a control strategy to address this problem using a decentralized heuristic method to estimate the presence of potentially fragile configurations. Based on this method, the authors propose a solution to mitigate such fragile configurations: adjusting the topology exploiting a robust control law that blends with other control objectives assigned to the multi-robot system.

This method was implemented on a real multi-robot system in [11], where the performance is evaluated considering an area coverage task, in the presence robotic failures and imperfect communication. The method proposed in [11] is a linear combination of different control laws and its overall performance heavily depends on the choice of weights, namely the gains (or hyper-parameters) assigned to each single control law. In [11], we exploited an offline optimization algorithm to automate the choice of such parameters, using preliminary experimental data. The main drawback of this solution is the fact that the optimal parameter choice is affected by the specific topology under consideration, thus making the offline process sub-optimal.

In this article, we experimentally evaluate the methodology proposed in [12]—an online optimization strategy that allows the multi-robot system to compute an optimal set of parameters during its mission, based on the current knowledge of the topology of the network. Our starting points are (i) the control law proposed in [10]—to improve the robustness of an initially connected multi-

robot topology—and (ii) the different fault-injection protocols descried in [11]. We combine and extend our previous work [12] to provide the following contributions: (i) simulations to compare, evaluate, and justify the choice of a scalarizing function for our multi-objective problem; (ii) real-life experiments with eight robots (K-team Khepera IV) and the injection of transient faults in the communication infrastructure; and finally (iii), real-life experiments with up to eight robots and the injection of permanent faults in the form of sudden, independently distributed hardware breakdowns. Results show that the proposed approach continues to perform as intended, even in the presence of such hazards.

The rest of this paper is organized as follows: Section 2 contextualizes our work among several other related contributions from recent years; we present background theory regarding network properties in Section 3; we discuss the multi-robot system model under evaluation in Section 4; and Section 5 outlines the proposed control architecture. Then, its integration with an online optimization strategy is described in Section 6, and we discuss our simulation results. In Section 7, we introduce our real-life robotic set-up, our experimental campaign, and we comment the obtained results . Finally, Section 8 concludes the article. Appendices A and B discuss about our choice of a scalarizing function and two fault-injection modes, respectively.

2. Related work

Swarm robotics is a research field that lies at the intersection of robotics and multi-agent systems and deals with large collections of relatively simple and mostly homogeneous, autonomous robots. Swarm intelligence [13], in particular, investigates the coordinated behaviours of these multi-agent systems, while swarm engineering [14] provides tools and methodologies to mimic them. In a recent perspective on Science Robotics, Yang et al. [1] listed the current "grand challenges" of robotics: these challenges include many of the issues we address in this work: "robot swarms", "exploration in extreme environments",

and "abilities to adapt, to learn, and to recover and handle failures". The growing interest of the research community for swarms and multi-robot systems has led to the introduction of many swarm-specifics tools, including simulators [15], programming languages [16, 17, 18], and design patterns [19, 20]—several of which we exploited in preparing this contribution.

When considering swarms, where each agent is a rather constrained sensing and computing platform, connectivity—and the ability for the robots to exchange information—is an important enabling property. Akram and Dagdeviren [21] used Steiner trees to address the "movement assisted connectivity restoration problem" and discovered it to be NP-Hard. Feng and Hu [22] studied connectivity-preserving rendez-vous accounting for battery levels and communication costs. Their proposed approach required to split the original task into sub-problems. Mosteo et al. [23] investigated the multi-robot routing problem under communication constraints and compared multiple algorithmic approaches (including greedy, TSP-based, and auction-based, ones), yet only through numerical simulations.

In the literature, connectivity maintenance methodologies draw inspiration from many different fields and theoretical frameworks. A large body of work belongs the area of wireless sensor networks (WSNs). These systems share several point of contact with networked multi-robots but differ mostly in the way they contemplate the mobility and reconfigurability of their nodes—often relegated to the design-time. Li et al. [24] review methodologies to compute the optimal density of the relay nodes in a WSN to ensure connectivity. Ghosh and Das [25] address the problem of WSN deployment to maximize coverage while maintaining connectivity. Jourdan and de Weck [26] apply a multi-objective Genetic Algorithm (GA) to optimize the layout of WSN, while Kulkarni and Venayagamoorthy [27] investigate the use of Particle Swarm Optimization (PSO). El-moukaddem et al. [28] study WSN with mobile nodes that can be used to optimize connectivity (without modifying the underlying network topology).

Narrowing our scope to multi-robot research, Friedman et al.[29] used Ant Colony Optimization for the "sometimes conflicting goals of fast travel time

and good network performance". Krupke et al. [30] proposed a heuristic, multi-component control law allowing robots to follow multiple leaders without breaking the robotic network. Panerati et al. [31] described the recursive creation of robotic chains using situated communication and a distance gradient. Banfi et al. [32] used Integer Linear Programming to optimally redeploy "a team of mobile robots acting as communication relays". The work of Majcherczyk et al. [33] aimed at constructing a logical tree topology and compared the performance of its outwards and inwards creation. Much of the work described up to this point, however, implements either centralized or heuristic, best effort approaches. For the sake of scalability and theoretical soundness, we base this work on algebraic connectivity, instead. In spectral graph theory, algebraic connectivity is a proxy measure for the connectedness of a network. Algebraic connectivity, despite representing a global property of a graph, can be estimated in distributed fashion using the Laplacian matrix of a graph.

Bertrand and Moonen [34] showed that the distributed computation of the second smallest eigenvalue of a Laplacian (i.e. algebraic connectivity, λ_2, or just λ) and associated eigenvector (i.e. the Fiedler vector) can be achieved using the power iteration method and normalization based on "cooperative diffusion". Sahai et al. [35] proposed a "wave propagation"-based approach and local fast Fourier transforms to compute all eigenvalues and local components of each eigenvectors of a graph. Di Lorenzo and Barbarossa [36] presented a stochastic power iteration method that allows each node to estimate algebraic connectivity and use it to adapt its own transmission power. Poonawala and Spong [7] studied the decentralized estimation of algebraic connectivity in strongly connected networks. Finally, Khateri et al. [9] compared local connectivity maintenance approaches (preserving all the initial links) and global connectivity maintenance approaches (preserving algebraic connectivity) to conclude that the first can be quicker and simpler yet the latter allow to cover larger workspaces.

Several approaches to improve inter-robot communication effectively exploit control strategies that maximize algebraic connectivity as a way to preserve connectedness. Ji and Egerstedt [8] proposed multiple nonlinear feedback laws

based on the Laplacian of a graph to solve the rendez-vous and formation-control problems while ensuring connectedness. De Gennaro and Jadbabaie [37] used an exponential decay model to characterize communication links and a potential-based control law that maximizes λ_2 through the supergradient method. Similarly, Yang et al. [3] and Sabattini et al. [4] implemented decentralized, power iteration-based estimation of λ_2 and gradient-based control. Robuffo Giordano et al. [38] enriched this class of control methodologies with the collision avoidance of static obstacles. Gasparri et al. [6] brought it to real-life experimentation with up to four robots. Yet, most of these works overlook certain subtleties required for robust real-world implementations, e.g., the presence of hard and soft errors, or adversarial behaviors.

In their review of fault-tolerance for robot swarms, Winfield and Nembrini [39] pointed out (i) motor failures and (ii) communications failures as hazard types number one and two (in a list of six). Robotic hardware failures and unreliable communication are, in fact, the non-idealities that we inject in our experiments (see Section B). Spanos and Murray [40] originally proposed a locally computable robustness metric called "the geometric connectivity robustness" and their work mostly revolved about modelling it in the context of a purely mathematical framework. Cheng and Wang [41] proposed a hierarchy-based method to "re-organize robot teams that require connectivity when robots fail". Using hierarchical graphs, however, can increase the approach's fragility towards the leaders' failures. Hollinger and Singh [42] took a completely different road and proposed a methodology that does not enforce continual connectivity but, rather, only periodic connectivity. Despite the real-world experiment and encouraging performance, this problem still turns out to be NP-Hard. In another alternative approach to a similar problem, Caccamo et al. [43] proposed a communication-aware motion planner "with autonomous repair of wireless connectivity". Yet, this work relies on the existence of fixed-location access points. Finally, it is worth mentioning the work of Gil et al. [44] as they observed that networks and multi-robot systems can be gravely disrupted by the Sybil attack and implemented a new algorihtm to sense spoofers using the physics of wireless

signals.

The contribution in this article stems from the theoretical work in [45, 10, 46] about the simultaneous control of connectivity (through λ_2) and robustness (of the multi-robot network towards faults). In [11, 47], we originally validated the control law in real robots and in presence of faults through the manual screening of many control gains combinations. In [12], we showed that the selection of these control gains can be delegated to autonomous, online optimization. Yet, we did not investigate the interplay of this level of autonomy with the error models in [11], in this work, we finally fill the gap. We do so by carrying the control and algorithms presented in [12] into the real robotic setup of [11]—including two types of fault-injection.

3. Preliminaries: network properties

Consider an undirected graph $\mathcal{G}$, where $\mathcal{V}(\mathcal{G})$ and $\mathcal{E}(\mathcal{G}) \subset \mathcal{V}(\mathcal{G}) \times \mathcal{V}(\mathcal{G})$ are the vertex set and the edge set, respectively. Moreover, let $W \in \mathbb{R}^{N \times N}$ be the weight matrix: each element w_{ij} is a positive number if an edge exists between nodes i and j, zero otherwise. Since $\mathcal{G}$ is undirected, then $w_{ij} = w_{ji}$.

Let $\mathcal{L} \in \mathbb{R}^{N \times N}$ be the Laplacian matrix of graph $\mathcal{G}$ and $D = \text{diag}\left(\{k_i\}\right)$ be the degree matrix, where k_i is the degree of the i-th node of the graph, i.e., $k_i = \sum_{j=1}^{N} w_{ij}$. The (weighted) Laplacian matrix of the graph is then defined as $L = D - W$.

The Laplacian matrix of an undirected graph $\mathcal{G}$ exhibits some remarkable properties regarding its connectivity [48]. Let λ_i, $i = 1, \ldots, N$ be the eigenvalues of the Laplacian matrix, then:

- The eigenvalues are real, and can be ordered such that $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$.
- Define now $\lambda = \lambda_2$. Then, $\lambda > 0$ if and only if the graph is connected. Therefore, λ is defined as the algebraic connectivity of the graph: in a weighted graph, λ is a non-decreasing function of each edge weight.

The algebraic connectivity is a good estimator of how well a graph is connected. While global connectivity is a Boolean property of a graph, larger values of λ indicate that the removal of more edges can be tolerated before a disconnection to occur. However, it cannot express the robustness of the graph topology to failures of elements with regard to connectivity maintenance, i.e., how much a graph can tolerate losing edges or vertices without fragmenting.

The robustness to failures is related to some topological properties of the interconnected graph, mainly the degree distribution. Some nodes play important roles in the topology formation, they are called central nodes. These nodes are crucial to the network communication and their failure will likely have a significant effect on the overall network connectivity. Therefore, the evaluation of the impact of central node failures on the network connectivity provides means to assess its robustness to failures.

In this direction, the robustness level proposed in [45] relies on the concept of betweenness centrality (BC) [49] for evaluating the network robustness. BC establishes higher scores for nodes that are contained in most of the shortest paths between every pair of nodes in the network. Thus, removing nodes according to their BC ranking—from highest to lowest values—might quickly lead to network fragmentation. The definition of the robustness level is:

Definition 1 (Robustness level [45]). *Consider a graph $\mathcal{G}$ with N nodes. Let $[v_1, \ldots, v_N]$ be the list of nodes ordered by descending value of BC. Let $\varphi < N$ be the minimum index $i \in [1, \ldots, N]$ such that, removing nodes $[v_1, \ldots, v_i]$ leads to fragmentation, that is, the graph including only nodes $[v_{\varphi+1}, \ldots, v_N]$ being disconnected. Then, the network* robustness level *of $\mathcal{G}$ is defined as:*

$$\Theta(\mathcal{G}) = \frac{\varphi}{N} \tag{1}$$

The robustness level thus defines the fraction of central nodes that need to be removed from the network to obtain a disconnected network. Small values of $\Theta(\mathcal{G})$ indicate that failures of a small fraction of nodes may fragment the network; consequently, increasing $\Theta(\mathcal{G})$ means increasing the resilience of the

network to failures. We observe that $\Theta(\mathcal{G})$ is only an estimation of how far the network is from getting disconnected w.r.t. fraction of nodes removed. In fact, it might be the case that different orderings of nodes with the same BC produce different values of $\Theta(\mathcal{G})$.

From a local perspective of robustness assessment, a heuristic to estimate the vulnerability of a node by means of the information acquired from its 1-hop and 2-hop neighbors was proposed in [45]. The vulnerability level takes into account the strength of a node's local connections: a node exhibiting weak local ties is more vulnerable to failures, whereas faults in its neighborhood may compromise its communication with the largest connected component of the network.

We summarize this vulnerability assessment as follows: let $d(v, u)$ be the shortest path between nodes v and u, i.e., the minimum number of edges that connect nodes v and u. Subsequently, define $\Pi(v)$ as the set of nodes that are at a minimum distance of at most 2-hops from v:

$$\Pi(v) = \{u \in V(G) : d(v, u) \leq 2\} \tag{2}$$

Moreover, let $|\Pi(v)|$ be the cardinality of $\Pi(v)$, and $\Pi_2(v) \subseteq \Pi(v)$ be the set of 2-hop neighbors of v that comprises only nodes whose shortest path from v is exactly equal to 2-hops, namely:

$$\Pi_2(v) = \{u \in V(G) : d(v, u) = 2\} \tag{3}$$

Larger values of d would lead to exponentially larger computational requirements that cannot be unjustified for an approximated approach.[1]

Now let $Path_\beta(v) \subseteq \Pi_2(v)$ be the set of v's 2-hop neighbors that are reachable through at most β paths, namely:

$$Path_\beta(v) = \{u \in \Pi_2(v) : L(v, u) \leq \beta\}, \tag{4}$$

[1]This heuristic was first proposed in [45] and validated in different scenarios, including network sizes, topologies, failure methodologies, model parameterization. The performance of information acquired from the 2-hop neighborhood was demonstrated to perform well not only for evaluating but also for mitigating the vulnerability of networks with respect to connectivity.

where $L(v,u)$ is the number of the shortest 2-hop paths between nodes v and u. Notice that β defines the threshold for the maximal number of paths between a node v and each of its u neighbors that are necessary to include u in $Path_\beta(v)$. Thus, setting a low value for β allows identifying fragile 2-hop neighbors connections.

Hence, the value of $|Path_\beta(v)|$ is an indicator of the magnitude of node fragility w.r.t. connectivity, and the vulnerability level of a node regarding failures is given by $P_\theta(v) \in (0,1)$:

$$P_\theta(v) = \frac{|Path_\beta(v)|}{|\Pi(v)|} \tag{5}$$

We will hereafter use $\beta = 1$, in order to identify 2-hop neighbors that are connected by a single path, which can represent a critical situation for network connectivity in scenarios of failures. A larger value of $P_\theta(v)$ increases the probability of a robot to set itself as vulnerable, thus improving its robustness.

4. System model and problem formulation

We assume a team of N mobile robots that are able to communicate with each other within a communication radius R, resulting in a communication topology represented by an undirected graph $\mathcal{G}$.

Let the state of each robot be its position $p_i \in \mathbb{R}^m$, and let $p = \left[p_1^T \dots p_N^T\right]^T \in \mathbb{R}^{Nm}$ be the state vector of the multi-robot system. Let each robot be modeled as a single integrator system, whose velocity can be directly controlled:

$$\dot{p}_i = u_i \tag{6}$$

where $u_i \in \mathbb{R}^m$ is a control input.

For each robot, the control input has to be designed so that a global objective can be accomplished. As a proof of concept, in the rest of the paper, we will refer to a scenario in which the robots are controlled to spread in a given area while avoiding collisions. However, the proposed methodology can be easily extended to other coordinated control objectives [47].

It is worth noting that coordinated objectives can be achieved only if information can be exchanged among the robots, that is, if the communication graph is connected and the robots keep this property as the system evolves. However, when considering real robotic systems, failures can not be neglected: robots may stop working unexpectedly and become unable to collaborate.

In this paper we combine three control laws, aiming at the achievement of a common objective (area coverage, in our case) while ensuring the collision avoidance and connectivity maintenance for the communication graph. The combination of the different control laws aims at maximizing a global performance index. This index defines a trade-off between the area actually covered by the robot and the level of connectivity of the communication network.

Note that connectivity is only guaranteed in free-fault environments because failures have an unpredictable nature and cascading failures can seriously damage the system connectivity. On the other hand, the mechanism for resilience improvement was demonstrated to be able to postpone or avoid network fragmentation, including cases where failures are concentrated over short time spans [10].

5. Overview of the control architecture

Referring to the kinematic model in Equation (6), in the following, we consider each robot to be controlled by means of a control input defined as the superposition of three different terms, that is:

$$u_i = \sigma_i u_i^c + \psi_i u_i^r + \zeta_i u_i^d \tag{7}$$

The components of the control inputs are defined as follows:

- The term $u_i^c \in \mathbb{R}^m$ represents the connectivity preservation control input. The role of this control input is to enforce that, if the communication graph is initially connected, then it will remain connected as the system evolves.

- The term $u_i^r \in \mathbb{R}^m$ represents the topology resilience improvement control input. This term aims at minimizing the impact of failure on the network connectivity by avoiding topological configurations that could induce a disconnection in the communication graph in case of failure of one or more robots.

- The term $u_i^d \in \mathbb{R}^m$ represents the desired control action. This encodes the coordinated objective that the multi-robot system needs to achieve. In this paper, we consider the objective to be the uniform coverage of a given area.

- The hyper-parameters $\sigma_i, \psi_i, \zeta_i \geq 0$ represent linear combination gains. They define the relative importance of the separate control laws.

It is worth noting that the overall behavior of the multi-robot system is defined by the way in which each individual control action is defined and by how they are combined. Indeed, a different choice of the linear combination gains leads to a different behavior of the multi-robot system.

In the following subsections, we introduce the individual control actions which are considered for implementation in the rest of the paper.

5.1. Connectivity preservation

The connectivity preservation control term u_i^c is designed, as in [4], to ensure that the value of the algebraic connectivity λ never goes below a given threshold $\epsilon > 0$. As in [4], the following energy function can be used for generating the decentralized connectivity maintenance control strategy:

$$V(\lambda) = \begin{cases} \coth(\lambda - \epsilon) & \text{if } \lambda > \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

The control law is designed to drive the robots to perform a gradient descent of $V(\cdot)$, which ensures preservation of the graph connectivity. Considering the robot model introduced in (6), the control law is defined as follows:

$$u_i = u_i^c = -\frac{\partial V(\lambda)}{\partial p_i} = -\frac{\partial V(\lambda)}{\partial \lambda}\frac{\partial \lambda}{\partial p_i}. \tag{9}$$

We observe that the connectivity preservation framework can be enhanced to consider also additional objectives. In particular, as shown in [38], the concept of generalized connectivity can be utilized to simultaneously guarantee connectivity maintenance and collision avoidance with environmental obstacles and among the robots.

5.2. Topology resilience improvement

The topology resilience improvement control term u_i^r is designed—in accordance with the methodology defined in [46, 10]—to drive the robots toward an improved resilience of the interconnection topology. Based on the concept of vulnerability level introduced in (5), this control strategy aims at increasing the number of links of a potentially vulnerable node i by driving it towards the barycenter of the 2-hop neighbors that are in $Path_\beta(i)$, thus decreasing its distance to them and eventually creating new edges in the communication graph. It is important to note that, if properly defined, $Path_\beta(i)$ contains the i's 2-hop neighbors with fragile connections.

Considering the robot model introduced in (6), the control law is defined as follows:

$$u_i^r = \xi_i \frac{x_\beta^i - p_i}{\left\| x_\beta^i - p_i \right\|} \alpha, \tag{10}$$

where $x_\beta^i \in \mathbb{R}^m$ is the barycenter of the positions of the robots in $Path_\beta(i)$ (see Equation (4) for its computation) and $\alpha \in \mathbb{R}^+$ is a scalar coefficient setting the velocity magnitude of each robot[2].

Parameter ξ_i takes into account the vulnerability state of a node i, i.e., $\xi_i = 1$ if node i identifies itself as vulnerable or $\xi_i = 0$ otherwise. As in [46, 10], we set as vulnerable those robots i exhibiting high values for $P_\theta(i)$: then, ξ_i is defined

[2]Pathological situations may exist in which (10) is not well defined, namely when $p_i = x_\beta^i$. However, this corresponds to the case where the i-th robot is exactly in the barycenter of its weakly connected 2-hop neighbors: in practice, this never happens when a robot detects itself as vulnerable.

as follows

$$\xi_i = \begin{cases} 1 & \text{if } P_\theta(i) > r \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where $r \in (0, 1)$ is a random number drawn from a uniform distribution, i.e., if $P_\theta(i) > r$, then the i-th robot considers itself as vulnerable. It is worth remarking that, according to (5), each robot can evaluate its vulnerability level in a decentralized manner.

5.3. Area coverage and collision avoidance

To control the robot to evenly spread over a given area while avoiding collisions, we propose to use the well-known control strategy based on the Lennard-Jones potential [14]. At distance x from its origin, the potential equation is:

$$P_{LJ} = \iota \left(\left(\frac{\delta}{x} \right)^a - 2 \cdot \left(\frac{\delta}{x} \right)^b \right) \tag{12}$$

When considering robot i and multiple neighboring robots j's ($\in \mathcal{N}(i)$), this entails that the desired control action equations can be written as:

$$u_i^d = -\iota \sum_{j \in \mathcal{N}(i)} \left(\left(\frac{a \cdot \delta^a}{x_{ij}^{a+1}} \right)^a - 2 \cdot \left(\frac{b \cdot \delta}{x_{ij}^{b+1}} \right)^b \right) \tag{13}$$

where parameters ι and δ define the potential function shape and x_{ij} is the inter-robot distance between i and j. Exponents a and b are set to 4 and 2. For the sake of collision avoidance, we set δ to be larger than the communication range of the robots.

6. Optimized control strategy

This section presents the methodology that we used to perform the online optimization of control gains σ_i, ψ_i, ζ_i introduced in Equation (7). The goal is to allow each robot to identify the most appropriate set of parameters as the system evolves.

The ideal performance is defined starting from the desired global behaviour, that is, achieving the largest area coverage while keeping a high level of connectivity. For this multi-objective problem, we define the following scalarizing function:

$$f_{obj}(t) = \lambda_2(t)\mathcal{A}(t) \tag{14}$$

where $\lambda_2(t)$ is the algebraic connectivity of the communication graph at time t, and $\mathcal{A}(t)$ is the value of the covered area at time t (see also Appendix A for discussion on alternative implementations of Equation (14)).

The choice of this scalarizing function is motivated by the fact that we are dealing with a multi-objective problem comprising two performance metrics with different domains and straightforward way to avoid an adaptive normalization scheme is to consider the metrics' product [50]. We also observe that the intent of our work is to optimize the control law for algebraic connectivity λ and area coverage $\mathcal{A}$ only. The robustness component u^r does not represent an objective *per se* but rather a hint to the multi-robot system to make it more robust and resilient once faults (imperfect communication and robotic failures) are injected.

Since (14) depends only on the actual position of the robots and not directly on the control gains, the predicted value of the scalarizing function at the next time step is considered in the formulation of the optimization process. Let consider the $j-th$ robot in the team, the solution of the constrained optimal control problem:

$$\begin{aligned} \max_{\sigma,\psi,\zeta} \quad & f_{obj}(t+\Delta t) \\ s.t. \quad & p_i(t+\Delta t) = p_i(t) + u_i(t)\Delta t \\ & u_i(t) = \sigma u_i^c + \psi u_i^r + \zeta u_i^d \\ & \sigma,\psi,\zeta \leq \Omega_{max} \\ & \|u_i\| \leq u_{max} \\ & i = 0, ..., N-1 \end{aligned} \tag{15}$$

returns the optimal set of gains $\sigma_j, \psi_j, \zeta_j$. $p_i(t)$ represents the position of the $i-th$ robot in the team available to robot j. With this knowledge, robot j computes $u_i(t)$, namely, the control input of the $i-th$ robot. Euler's method is

then used to estimate the future positions of the robots in the team $p_i(t + \Delta t)$, exploiting the starting positions $p_i(t)$, the control inputs $u_i(t)$ and the step time Δt. Ω_{max} represents the maximum value of the gains while u_{max} represents the maximum control input. With a simplifying assumption, each robot in the team solves the optimization problem under the hypothesis that all the robots will move using the same set of gains.

6.1. Optimization algorithms

We are now left with the task of selecting an optimization methodology that can allow us to find the ideal combination of the gains σ, ψ, ζ such that the objective function introduced in (14) is maximized. We observe that the scalarizing function we selected (according to the considerations made in Appendix A) is the product of nonlinear functions, that is, algebraic connectivity λ_2 (the computation of the eigenvalues of the Laplacian matrix is nonlinear) and area coverage (the sum of the non-overlapping portions of the disks around each robot).

Consequently, we searched among optimization methods that are not too computationally expensive but also well suited for such nonlinear problems. We evaluated the following approaches [51]:

- Grid search optimization provides a uniform and homogeneous screening of the parameters space. The main advantage of this method is the accuracy of the solution, which can be freely refined if one is not constrained by the computational time requirements.
- Random search optimization. A probabilistic search does not require the gradient of the objective function and can tackle non-continuous or non-differentiable objective functions. The optimal set of parameters is found by probing the domain space with a uniform probability distribution. Heuristic and random search algorithms can provide a lower computational burden at the cost of relinquishing guarantees of optimality.

- The augmented Lagrangian optimization algorithm is especially suited for constrained optimization problems, it requires to (i) first penalize the objective function, (ii) translate the constrained optimization problem into a series of unconstrained problems, and then (iii) adds a term designed to mimic a Lagrange multiplier and improve precision and convergence speed. The algorithm uses the gradient of the objective function. In the case of Equation (14), numerical differentiation is exploited.

6.2. Implementation and evaluation

As we want to compare the optimization algorithms from Subsection 6.1 both in terms of quality of the solution and computational requirement, we implemented the following simulated experiments. Eight robots are placed in a squared arena. Positions of all the robots are shared with all the other robots. As we are in a non-fully connected network, we use a consensus mechanism—i.e., virtual stigmergy [19]. Using this shared knowledge, each robot computes the components of the control input of every robot in the team (u_i^c, u_i^r and u_i^d in Equation (7)).

We define as $O_p \in \mathbb{Z}^+$ the optimization period and $G_p \in \mathbb{Z}^+$ the number of generated points. Every O_p control steps, every robot optimizes and updates its own set of gains to be used in (7) as follows:

1. A maximum of G_p gains—tuples $\langle \psi, \sigma, \zeta \rangle$—are generated by the optimization algorithm (one of those described in Subsection 6.1).
2. For each tuple, the robots (i) predicts the positions of all other robots at the subsequent time step integrating (7) and (ii) evaluate the objective function introduced in Equation (14).
3. The gains returning the greater evaluation of Equation (14) are selected (note that, due to asynchronicity, imperfect communication, and the random nature of one of the proposed optimization approaches, there could be different gains for different robots, as shown in Figure 4).

The optimization period O_p is set by the user for all the optimization methods. The number of generated points G_p can be set by the user for the grid

search optimization and for the random search optimization algorithms. For the augmented Lagrangian optimization algorithm, the value of G_p is determined by the convergence criteria of the algorithm itself.

These steps were implemented using the Buzz scripting language [16], and simulations were run using the multi-physics environment of ARGoS [15]. We evaluated the performance of the three optimization methodologies in a network of eight two-wheeled robots and we compared it against the same robot team using constant gains. We screened the Cartesian product (i.e., all combinations) of the following gain assignments:

$$\psi = \{0, 1, 2\} \quad \sigma = \{0, 1, 2\} \quad \zeta = \{0, 1\} \tag{16}$$

The results of these simulations are summarized in Table 1 and Figure 1, which presents the evolution of the objective function (14) as the experiments progress. The three colored lines represents, respectively, the objective function values obtained by each of the optimization algorithm, while the black line with the grey shadow represent the average value and standard deviation of the objective function provided by the screened set of constant gains. Unsurprisingly, the value of the objective function is typically greater when using an optimization method (with respect to constant gains). Figure 1 also shows that random search optimization performs significantly on-par or better than other methods. This result can be explained by the fact that the search space $[0, 10]^3$ is not highly dimensional nor particularly complex. As the computational requirements of a random search are generally modest, we choose it as the preferred optimization algorithm for the rest of this work. We then performed a second set of simulations to investigate how the choice of the hyper-parameters G_p and O_p influences the optimization performance. We run simulations for $G_p = \{250, 400, 2200, 4000\}$ and $O_p = \{1, 10, 50\}$.

The results obtained from these simulations are presented in Figure 2. We observe that different parameter choices provide similar and often comparable results, as quantified by the objective function. Hence, we reckon that the opti-

Algorithm	Objective function	Topology evolution	Computational time
Augmented lagrangian	=	=	=
Random search	↑	=	=
Grid search	↓	=	↓

Table 1: Comparative summary of the optimization algorithms described in Subsection 6.1

mization algorithm can be effectively run using a limited number of generated points (i.e. $G_p = 250$) and sporadic optimization (i.e. $O_p = 50$) to reduce the computational requirements without hurting the overall performance of the multi-robot team. Having selected random optimization with $O_p = 50$ and $G_p = 250$, we run an extensive simulation campaign whose results are reported and discussed in section 6.3

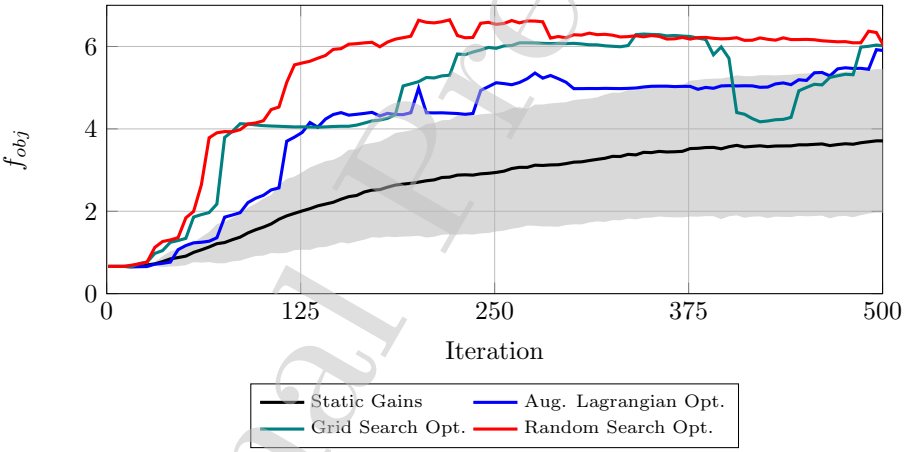


Figure 1: Objective function evolution comparison: static gains versus optimized gains when using augmented Lagrangian, random, and grid searches. For the Grid search optimization and for the Random search optimization algorithm, O_p and G_p are set to 1 and 4000 respectively. For the augmented Lagrangian optimization algorithm, the value of O_p is set to 1 while G_p is determined by the convergence criteria.

Eighty additional simulations were also performed in order to assess scalability of online optimization when the number of robots in the team changes. We performed simulation with 3,4,6 and 8 robots, $O_p \in \{1, 50\}$ and $G_p \in \{250, 400\}$.

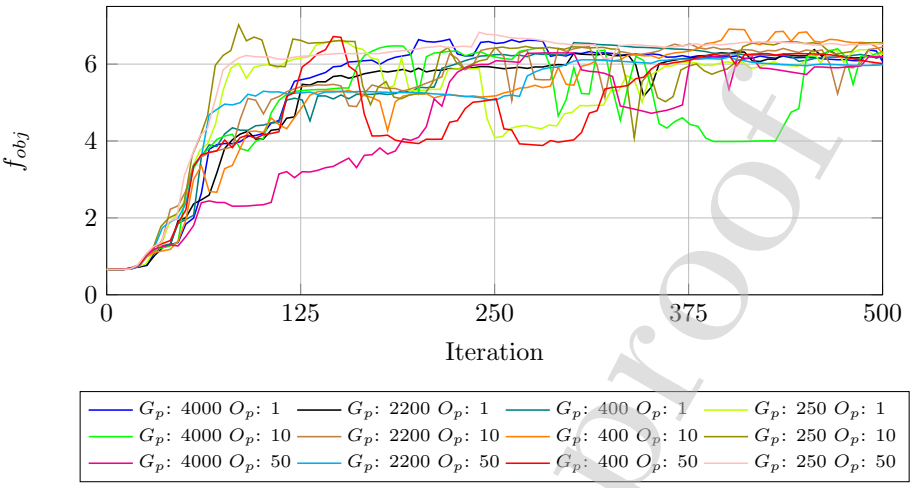


Figure 2: Evolution of the objective function f_{obj} (throughout the course of a 500-iteration simulation) when using random search optimization. Comparison of different hyper-parametrizations in terms of number of generated points G_p and optimization frequency O_p.

6.3. Simulations results and discussion

The results obtained performing the first simulations illustrated in section 6.2 are summarized in Figure 3, that shows the evolution of the main metrics [3], and Figure 4, that shows how the gains σ_i, ψ_i, and ζ_i evolve on-board each robot. Simulations were performed in a Fault-Free scenario and introducing the two fault-injection presented in Appendix B.

As expected, in the Fault-free scenario the objective function increase during the simulation while the algebraic connectivity and the covered area reach a trade-off. A good performance can also be observed for the robustness point of view, that increases over time also if it is not considered in optimization. This is mainly due to the presence of the term u_i^r in the control law. Similar

[3] Note that the second and third column of Figures 3 and 4 present results contemplating the fault-injection protocols introduced in Appendix B.

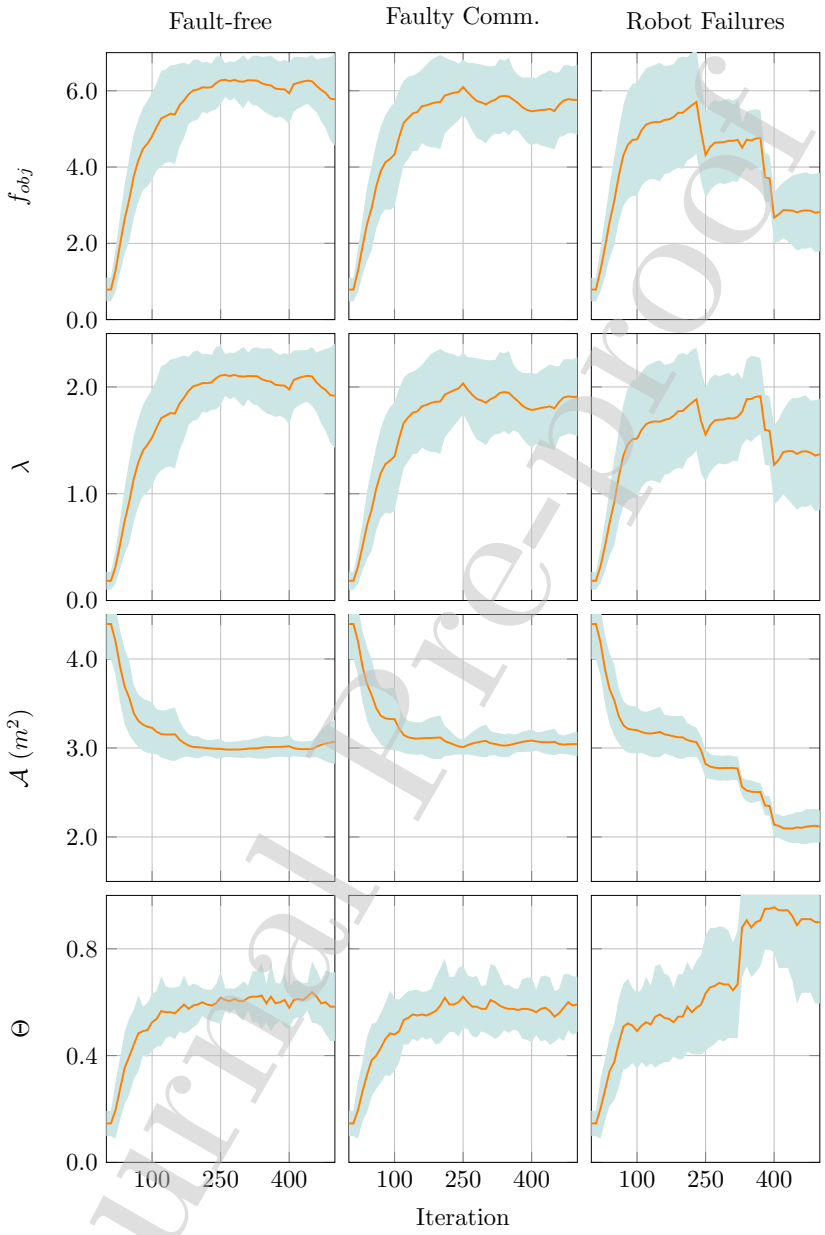


Figure 3: Scalarizing/objective function f_{obj}, algebraic connectivity λ, area coverage $\mathcal{A}$ (in m^2), and robustness Θ in different simulations scenarios (fault-free, with faulty communication, and with hardware failures). Simulations were repeated 30 times, over 500 ARGoS simulator iterations, in each fault-injection scenario. The orange line and teal shadow report average and standard deviation, respectively.

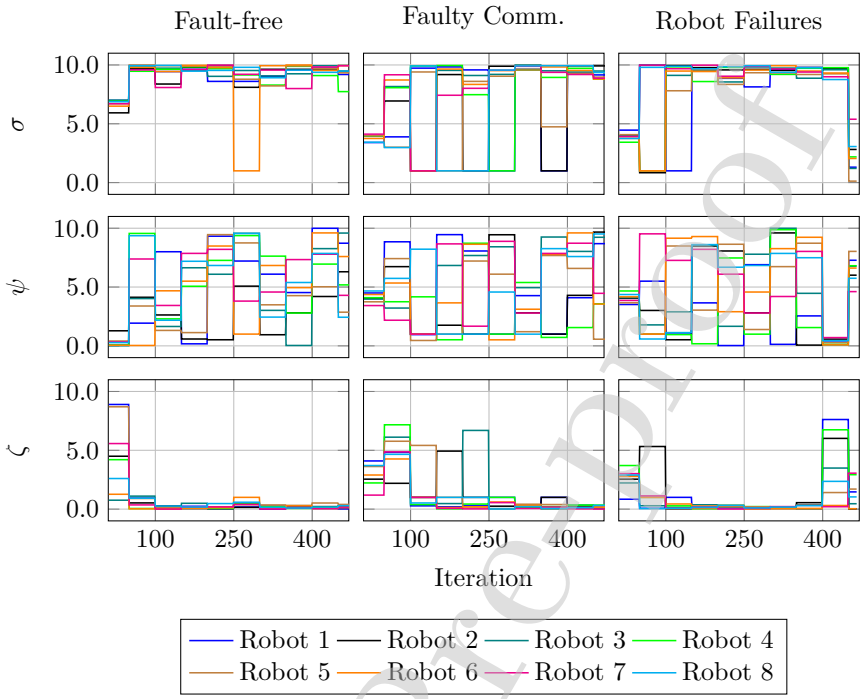


Figure 4: Evolution of control law (7) gains σ, ψ, and ζ, for each of the 8 robots, in different simulations scenarios (fault-free, with faulty communication, and with hardware failures), for a fixed starting configuration. Simulations were run over 500 ARGoS simulator iterations, in each fault-injection scenario. Optimization was based on the random search approach using $O_p = 50$ and $G_p = 250$.

Optimization	**# of Robots**			
Parameters	**3**	**4**	**6**	**8**
$O_p = 1$, $G_p = 400$	126, 132	373, 231	642, 444	928, 420
$O_p = 50$, $G_p = 400$	146, 157	466, 300	714, 389	938, 430
$O_p = 1$, $G_p = 250$	149, 160	454, 289	721, 462	982, 509
$O_p = 50$, $G_p = 250$	132, 144	471, 344	828, 502	759, 284

Table 2: Percentage (%) increase of the objective function f_{obj} value between the start and the end of 500-iteration simulations. The values show the average value and standard deviation of multiple simulations varying the optimization hyper-parameters (the rows) and the number of robots (the columns) in the team.

performances can also be appreciated for both the fault-injection scenario. In particular, we can observe a decrease in the objective function in the case of robot failures, associated to the decrease in the number of robots in the team. The same considerations are confirmed by the low value of ζ and the high value of σ and ψ for all the simulation and for all the faults scenario. The results obtained for the scalability simulation campaign are reported in table 2 that reports the average value and the standard deviation of the percentage increase of the objective function value between the start and the end of the experiment. One can observe larger percentage increases as the number of robots goes up. This is expected since the number of robots leads to an inevitable increase in the covered area, while the absolute value of algebraic connectivity is not significantly affected by the number of robots for teams of this size. Nonetheless, Table 2 show how the optimizer performs as intended independently of the number of robots in the team and its hyper-parameters.

7. Experimental validation

Transitioning from simulation to real robots can be challenging and results in performance degradation, especially with resource constraint hardware [11]. To demonstrate the portability of the proposed online optimization, and to analyze how hardware limitations affect the choice of the optimization parameters (i.e., the generated points G_p and optimization period O_p), we used an actual distributed multi-robot system to test our methodology. The robot team consists of eight two-wheeled differential-drive K-Team Khepera IV shown in Figure 5. Each robot is equipped with an 800MHz ARM Cortex-A8 and the linux-based Yocto operating system[4].

A camera-based tracking system consisting of four OptiTrack[5] Prime13 cameras (see Figure 5), and the `blabbermouth`[6] communication software are com-

[4] `https://www.k-team.com/mobile-robotics-products/khepera-iv`
[5] `https://optitrack.com/products/prime-13/specs.html`
[6] `https://github.com/MISTLab/blabbermouth`

bined to emulate range and bearing sensors for each robot. The communication infrastructure is based on traditional Wi-Fi and, integrating the information from the camera-based positioning system, we emulate communication ranges up to a fixed distance $R = 60cm$ (analogue to the setup used in [11]). All information on-board each robot is in local coordinates. OptiTrack sends to every robot the positions of its neighbors in its own local coordinates. The messages that robots send to each other also use the robots' own coordinate system (and, thus, they have to be transformed on board each receiving robot).

Figure 5: One of four OptiTrack Prime 13 cameras and one of eight K-Team's Khepera IV robots ($\varnothing = 14.0\,cm, h = 6.0\,cm$) used for the experimental setup in Section 7.

The optimization procedure described in Section 6.2 is embedded into the Khepera IV-specific virtual machine `bzzkh4`[7] that is used to execute the Buzz byte code of each robotic controller. Using the parameters studied in simulation as a starting point, we determined the optimization times Δ_t for the on-board processing at the varying of G_p. We obtain Δ_t's of $8'41''$, $46'47''$ and $84'23''$ as runtimes for 400, 2200 and 4000 generated points G_p, respectively. That is, with increasing G_p, Δ_t increases linearly and ranges from minutes to hours. Considering these computational demands, it is sensible to run the online optimization on the Khepera IV every $O_p = 50$ steps with a G_p of 250 points ($\Delta_t \sim 2'$). Simulations and experimental validation iterate over a fix number of control steps. The duration of each experiment is set to 500 and 300 such iterations, respectively, and every experiment was repeated starting from four, randomly selected initial poses. Due to the potentially varying processing times on each robot, the team of Khepera IVs operates asynchronously.

[7] `https://github.com/MISTLab/BuzzKH4`

7.1. Experimental results and discussion

The results obtained combining the robotic set-up described in Section 7 and the two fault-injection protocols presented in Appendix B are shown in Figure 6. The three columns of Figure 6 refer to the three different fault scenarios: the absence of faults (left), the injection of faults in the communication layer (centre), and the injection of faults in the robotic hardware (right). The four rows present the evolution of different metrics, namely the scalarizing function f_{obj}, algebraic connectivity λ, area coverage $\mathcal{A}$, and robustness Θ. Each plot displays an average value (the orange line) and a standard deviation (the teal shade) computed over the repeated experiments conducted from different initial poses.

The leftmost column in Figure 6 presents our baseline performance for the optimized control law. The fault-free results resembles, in fact, those in [11]—where the choice of gains $\langle \psi, \sigma, \zeta \rangle$ was surrendered to manual screening. Once again, we can observe a natural trade-off between the values of λ, Θ and $\mathcal{A}$. The most notable result in Figure 6 certainly comes from the central column. Here, we clearly see how static gains [11] and online optimization produce very different results. In [11], we had noted that the presence of faulty communication could lead the robots to favour λ over $\mathcal{A}$, resulting in more compact formations. In Figure 6, this behaviour is remarkably not present and—albeit deteriorated w.r.t. the fault-free scenario—both λ and $\mathcal{A}$ increase over time. In fact, the online adjustment of the control gains appears to facilitate the balance between the two objectives. Finally, in the rightmost column, we observe that, in presence of hardware failures, $\mathcal{A}$ is predictably and inevitably weakened. Yet, both λ and Θ can be driven up by the proposed approach (note that the larger absolute values are justified by the fact that they refer to progressively smaller networks, with less than eight robots).

Table 3 summarizes the results of nine two-tailed, paired t-tests between the initial and final distributions of metrics f_{obj}, λ, and $\mathcal{A}$ using the data from Figure 6. These suggest that the samples for all three metrics have distribution with different means, i.e., the proposed approach drives them towards the desired topology, even in of the presence of faulty communication. The results

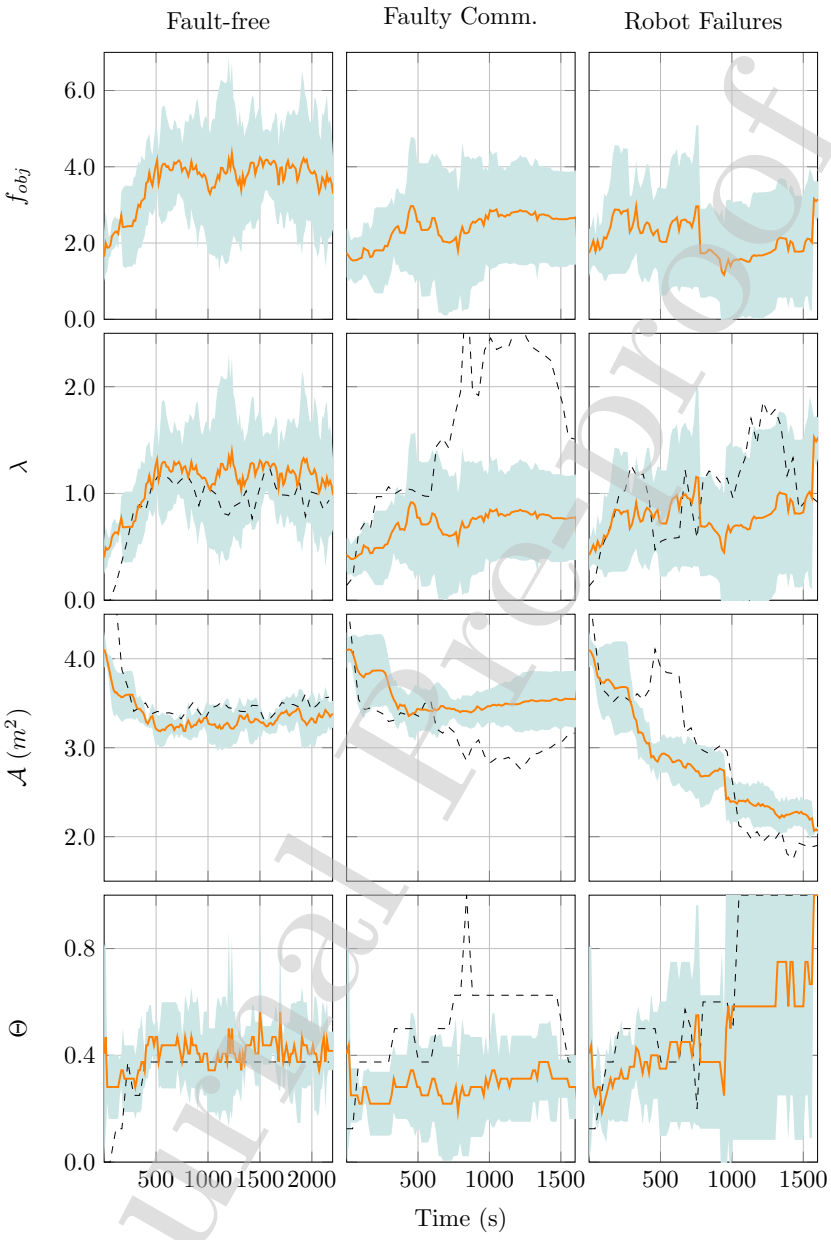


Figure 6: Scalarizing/objective function f_{obj}, algebraic connectivity λ, area coverage $\mathcal{A}$ (in m^2), and robustness Θ in different experimental scenarios (fault-free, with faulty communication, and with hardware failures), as observed by the OptiTrack tracking system. The orange line and teal shadow report average and standard deviation, respectively. The dashed black line shows the performance (from [11]) of static gains $\langle \psi : 1, \sigma : 2, \zeta : 1 \rangle$.

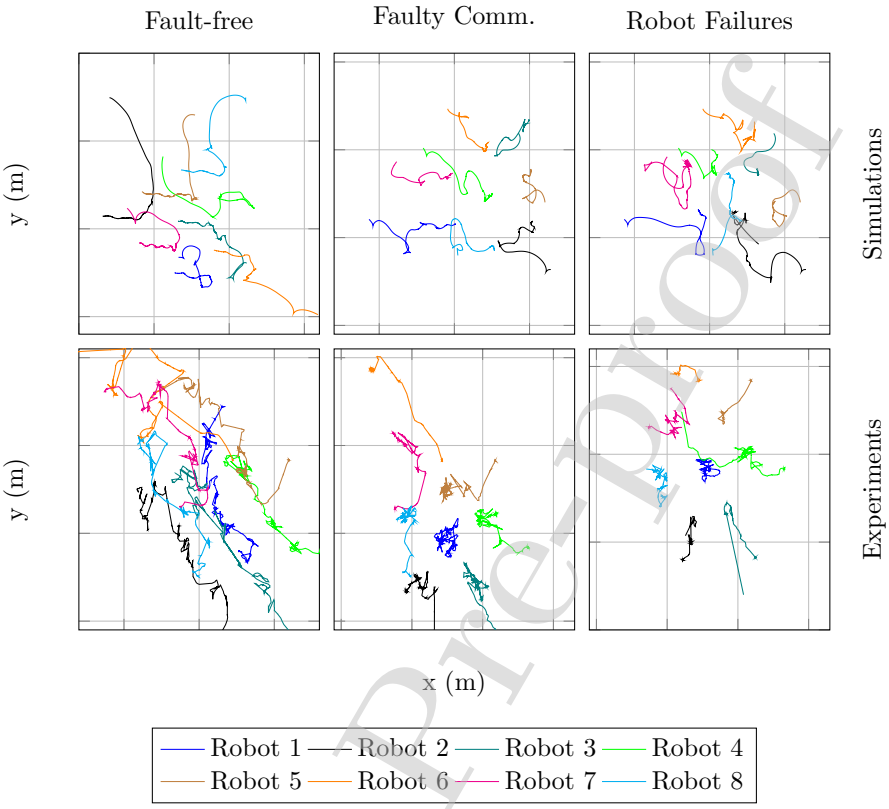


Figure 7: Comparison of trajectory traces in simulations and real-life experiments for different fault-injection scenarios (fault-free, with faulty communication, and with hardware failures).

in the presence of robotic failures, on the other hand, are not equally clear-cut. We performed the same t-tests using the data from Figure 3 to confirm that simulations with robotic failures lead to significantly different means for all three metrics. Finally, Figure 7 compares the trajectory traces of the robots in simulations and experiments in the different fault-injection scenarios.

8. Conclusions

In this article, we experimentally evaluated the methodology proposed in [12] (i.e., the online optimization of resilient multi-robot networks) against faults. Our starting points were (i) the control law proposed in [10]—to improve the

	Fault-free	Faulty Comm.	Robot Failures
f_{obj}	0.1127	0.3979	0.9419
λ	0.0952	0.2870	0.3135
$\mathcal{A}$	0.0445	0.0493	0.0005

Table 3: Two-tailed, paired t-tests between initial ($t = 0s$) and final ($t = 2000s$ for the fault-free scenario, $1500s$ for the faulty communication and robot failures scenarios) distributions of the data from figure 6. Smaller values indicates that one should be more inclined to reject the null hypothesis (of the samples coming from distributions with equal mean). The same t-tests for the data from Figure 3 all returned values ~ 0.

robustness of an initially connected multi-robot topology—and (ii) the different fault-injection protocols descried in [11]. We combined and extended all of our previous work to provide the following contributions: (i) simulations to compare, evaluate, and justify the choice of a scalarizing function for our multi-objective problem—that is, the simultaneous maximization of algebraic connectivity and area coverage; (ii) real-life experiments with eight robots (K-team Khepera IV) and the injection of transient faults in the communication infrastructure; and finally (iii), real-life experiments with up to eight robots and the injection of permanent faults in the form of sudden, independently distributed hardware breakdowns. The new experiments reveal that the proposed control strategy is, in fact, effective in improving coverage, connectivity, and robustness of a robot-team. Unlike static hyper-parameterization [11], online optimization proved to be effective in balancing conflicting goals in the presence of faulty communication. In the upcoming future, we intend to extend our work on connectivity and fault-tolerance even further to account for more sophisticated exploration strategies such as the use a "Voronoi tessellation"-based coverage contribution [47], a full-fledged distributed path planner, and study the existence of formal guarantees on robustness and connectivity maintenance for specific implementations of the u^d control contribution.

References

[1] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, et al., The grand challenges of science robotics, Science Robotics 3 (14).

[2] A. Prorok, M. A. Hsieh, V. Kumar, Adaptive distribution of a swarm of heterogeneous robots, Acta Polytechnica 56 (1) (2016) 67–75.

[3] P. Yang, , R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, R. Sukthankar, Decentralized estimation and control of graph connectivity for mobile sensor networks, Automatica 46 (2) (2010) 390–396.

[4] L. Sabattini, N. Chopra, C. Secchi, Decentralized connectivity maintenance for cooperative control of mobile robotic systems, I. J. Robotic Res. 32 (12) (2013) 1411–1423. doi:10.1177/0278364913499085.

[5] L. Sabattini, C. Secchi, N. Chopra, A. Gasparri, Distributed control of multi-robot systems with global connectivity maintenance, IEEE Transactions on Robotics 29 (5) (2013) 1326–1332.

[6] A. Gasparri, L. Sabattini, G. Ulivi, Bounded control law for global connectivity maintenance in cooperative multi-robot systems, IEEE Transactions on Robotics 33 (3) (2017) 700–717. doi:10.1109/TRO.2017.2664883.

[7] H. A. Poonawala, M. W. Spong, Decentralized estimation of the algebraic connectivity for strongly connected networks, in: American Control Conference (ACC), IEEE, 2015, pp. 4068–4073.

[8] M. Ji, M. Egerstedt, Distributed coordination control of multiagent systems while preserving connectedness, IEEE Transactions on Robotics.

[9] K. Khateri, M. Pourgholi, M. Montazeri, L. Sabattini, A comparison between decentralized local and global methods for connectivity maintenance of multi-robot networks, IEEE Robotics and Automation Letters 4 (2) (2019) 633–640. doi:10.1109/LRA.2019.2892552.

[10] C. Ghedini, C. Ribeiro, L. Sabattini, Toward fault-tolerant multi-robot networks, Networks 70 (4) (2017) 388–400. doi:10.1002/net.21784.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21784

[11] J. Panerati, M. Minelli, C. Ghedini, L. Meyer, M. Kaufmann, L. Sabattini, G. Beltrame, Robust connectivity maintenance for fallible robots, Autonomous Robots.
URL https://doi.org/10.1007/s10514-018-9812-8

[12] M. Minelli, M. Kaufmann, J. Panerati, C. Ghedini, G. Beltrame, L. Sabattini, Stop, think, and roll: Online gain optimization for resilient multi-robot topologies, in: N. Correll, M. Schwager, M. Otte (Eds.), Distributed Autonomous Robotic Systems, Springer International Publishing, Cham, 2019, pp. 357–370.

[13] E. Şahin, S. Girgin, L. Bayindir, A. E. Turgut, Swarm Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, Ch. 3, pp. 87–100. doi:10.1007/978-3-540-74089-6_3.

[14] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 (1) (2013) 1–41. doi:10.1007/s11721-012-0075-2.

[15] 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, M. Dorigo, Argos: a modular, parallel, multi-engine simulator for multi-robot systems, Swarm Intelligence 6 (4) (2012) 271–295. doi:10.1007/s11721-012-0072-5.
URL https://doi.org/10.1007/s11721-012-0072-5

[16] C. Pinciroli, G. Beltrame, Swarm-oriented programming of distributed robot networks, Computer 49 (12) (2016) 32–41.

[17] M. P. Ashley-Rollman, P. Lee, S. C. Goldstein, P. Pillai, J. D. Campbell, A language for large ensembles of independently executing nodes, in: P. M.

Hill, D. S. Warren (Eds.), Logic Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 265–280.

[18] K. Dantu, B. Kate, J. Waterman, P. Bailis, M. Welsh, Programming micro-aerial vehicle swarms with karma, in: Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, SenSys '11, ACM, New York, NY, USA, 2011, pp. 121–134. doi:10.1145/2070942.2070956. URL http://doi.acm.org/10.1145/2070942.2070956

[19] C. Pinciroli, A. Lee-Brown, 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 (Formerly BIONETICS), BICT'15, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2016, pp. 287–294. doi:10.4108/eai.3-12-2015.2262503.

[20] J. Bachrach, J. Beal, J. McLurkin, Composable continuous-space programs for robotic swarms, Neural Computing and Applications 19 (6) (2010) 825–847. doi:10.1007/s00521-010-0382-8.

[21] V. K. Akram, O. Dagdeviren, On hardness of connectivity maintenance problem in drone networks, in: 2018 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 2018, pp. 1–5. doi:10.1109/BlackSeaCom.2018.8433713.

[22] Z. Feng, G. Hu, A distributed constrained optimization approach for spatiotemporal connectivity-preserving rendezvous of multi-robot systems, in: 2018 IEEE Conference on Decision and Control (CDC), 2018, pp. 987–992. doi:10.1109/CDC.2018.8619773.

[23] A. R. Mosteo, L. Montano, M. G. Lagoudakis, Multi-robot routing under limited communication range, in: 2008 IEEE International Conference on Robotics and Automation, 2008, pp. 1531–1536. doi:10.1109/ROBOT.2008.4543419.

[24] J. Li, L. L. Andrew, C. H. Foh, M. Zukerman, H.-H. Chen, Connectivity, coverage and placement in wireless sensor networks, Sensors 9 (10) (2009) 7664–7693. doi:10.3390/s91007664.
URL https://www.mdpi.com/1424-8220/9/10/7664

[25] A. Ghosh, S. K. Das, Coverage and connectivity issues in wireless sensor networks: A survey, Pervasive and Mobile Computing 4 (3) (2008) 303 – 334. doi:https://doi.org/10.1016/j.pmcj.2008.02.001.
URL http://www.sciencedirect.com/science/article/pii/S1574119208000187

[26] D. B. Jourdan, O. L. de Weck, Layout optimization for a wireless sensor network using a multi-objective genetic algorithm, in: 2004 IEEE 59th Vehicular Technology Conference. VTC 2004-Spring (IEEE Cat. No.04CH37514), Vol. 5, 2004, pp. 2466–2470 Vol.5. doi:10.1109/VETECS.2004.1391366.

[27] R. V. Kulkarni, G. K. Venayagamoorthy, Particle swarm optimization in wireless-sensor networks: A brief survey, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 41 (2) (2011) 262–267. doi:10.1109/TSMCC.2010.2054080.

[28] F. El-Moukaddem, E. Torng, G. Xing, Mobile relay configuration in data-intensive wireless sensor networks, IEEE Transactions on Mobile Computing 12 (2) (2013) 261–273. doi:10.1109/TMC.2011.266.

[29] A. Fridman, S. Weber, V. Kumary, M. Kam, Distributed path planning for connectivity under uncertainty by ant colony optimization, in: 2008 American Control Conference, 2008, pp. 1952–1958. doi:10.1109/ACC.2008.4586778.

[30] D. Krupke, M. Ernestus, M. Hemmer, S. P. Fekete, Distributed cohesive control for robot swarms: Maintaining good connectivity in the presence of exterior forces, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 413–420. doi:10.1109/IROS.2015.7353406.

[31] J. Panerati, L. Gianoli, C. Pinciroli, A. Shabah, G. Nicolescu, G. Beltrame, From swarms to stars: Task coverage in robot swarms with connectivity constraints, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 7674–7681. doi:10.1109/ICRA.2018.8463193.

[32] J. Banfi, N. Basilico, S. Carpin, Optimal redeployment of multirobot teams for communication maintenance, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 3757–3764. doi:10.1109/IROS.2018.8593532.

[33] N. Majcherczyk, A. Jayabalan, G. Beltrame, C. Pinciroli, Decentralized connectivity-preserving deployment of large-scale robot swarms, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 4295–4302. doi:10.1109/IROS.2018.8594422.

[34] A. Bertrand, M. Moonen, Distributed computation of the fiedler vector with application to topology inference in ad hoc networks, Signal Processing 93 (5) (2013) 1106 – 1117. doi:http://dx.doi.org/10.1016/j.sigpro.2012.12.002.

[35] T. Sahai, A. Speranzon, A. Banaszuk, Hearing the clusters of a graph: A distributed algorithm, Automatica 48 (1) (2012) 15–24. doi:10.1016/j.automatica.2011.09.019.
URL http://dx.doi.org/10.1016/j.automatica.2011.09.019

[36] P. Di Lorenzo, S. Barbarossa, Distributed estimation and control of algebraic connectivity over random graphs, IEEE Transactions on Signal Processing 62 (21) (2014) 5615–5628. doi:10.1109/TSP.2014.2355778.

[37] M. C. De Gennaro, A. Jadbabaie, Decentralized control of connectivity for multi-agent systems, in: Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 3628–3633. doi:10.1109/CDC.2006.377041.

[38] P. Robuffo Giordano, A. Franchi, C. Secchi, H. H. Bülthoff, A passivity-based decentralized strategy for generalized connectivity maintenance, The International Journal of Robotics Research 32 (3) (2013) 299–323.

[39] A. F. T. Winfield, J. Nembrini, Safety in numbers: Fault tolerance in robot swarms, International Journal on Modelling Identification and Control 1 (1) (2006) 30–37.
URL http://infoscience.epfl.ch/record/100088

[40] D. P. Spanos, R. M. Murray, Motion planning with wireless network constraints, in: Proceedings of the 2005, American Control Conference, 2005., 2005, pp. 87–92 vol. 1. doi:10.1109/ACC.2005.1469913.

[41] L. Cheng, Y.-J. Wang, Fault tolerance for communication-based multirobot formation, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), Vol. 1, 2004, pp. 127–132 vol.1. doi:10.1109/ICMLC.2004.1380629.

[42] G. Hollinger, S. Singh, Multi-robot coordination with periodic connectivity, in: 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 4457–4462. doi:10.1109/ROBOT.2010.5509175.

[43] S. Caccamo, R. Parasuraman, L. Freda, M. Gianni, P. Ögren, Rcamp: A resilient communication-aware motion planner for mobile robots with autonomous repair of wireless connectivity, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 2010–2017. doi:10.1109/IROS.2017.8206020.

[44] S. Gil, S. Kumar, M. Mazumder, D. Katabi, D. Rus, Guaranteeing spoof-resilient multi-robot networks, Auton. Robots 41 (6) (2017) 1383–1400. doi:10.1007/s10514-017-9621-5.

[45] C. Ghedini, C. Secchi, C. H. C. Ribeiro, L. Sabattini, Improving robustness in multi-robot networks, in: Proceedings of the IFAC Symposium on Robot Control (SYROCO), Salvador, Brazil, 2015, pp. 63–68.

[46] C. Ghedini, C. H. C. Ribeiro, L. Sabattini, Toward efficient adaptive adhoc multi-robot network topologies, Ad Hoc Networks 74 (2018) 57 – 70. doi:10.1016/j.adhoc.2018.03.012.

[47] L. Siligardi, J. Panerati, M. Kaufmann, M. Minelli, C. Ghedini, G. Beltrame, L. Sabattini, Robust area coverage with connectivity maintenance, in: 2019 IEEE International Conference on Robotics and Automation (ICRA), 2019, pp. –.

[48] C. Godsil, G. Royle, Algebraic Graph Theory, Springer, 2001.

[49] S. Wasserman, K. Faust, D. Iacobucci, Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences), Cambridge University Press, 1994.

[50] J. Panerati, G. Beltrame, A comparative evaluation of multi-objective exploration algorithms for high-level design, ACM Trans. Des. Autom. Electron. Syst. 19 (2) (2014) 15:1–15:22.

[51] M. Avriel, Nonlinear programming: analysis and methods, Courier Corporation, 2003.

[52] E. W. Dijkstra, Selected Writings on Computing: A Personal Perspective, Springer-Verlag, Berlin, Heidelberg, 1982.

[53] A. Avizienis, J. C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transactions on Dependable and Secure Computing 1 (1) (2004) 11–33. doi:10.1109/TDSC.2004.2.

[54] J. Panerati, S. Abdi, G. Beltrame, Balancing system availability and lifetime with dynamic hidden markov models, in: 2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2014, pp. 240–247. doi:10.1109/AHS.2014.6880183.

[55] E. A. Elsayed, Reliability Engineering, 2nd Edition, Wiley Publishing, 2012.

A. Alternative scalarizing functions

To evaluate the impact and effectiveness of our scalarizing function choice on the overall system performance, we also run multiple simulations using the following arithmetic combinations of the two performance metrics:

1. The product of the performance metrics λ_2 and $\mathcal{A}$:

$$f_{obj} = \lambda_2(t) \cdot \mathcal{A}(t) \tag{17}$$

2. The sum of the performance metrics λ_2 and $\mathcal{A}$:

$$f_{obj} = \lambda_2(t) + \mathcal{A}(t) \tag{18}$$

3. The normalized sum of the performance metrics (with $\lambda_{2-tar.}$ and $\mathcal{A}_{tar.}$ set to 2.0 and 5.0, respectively, after preliminary evaluation):

$$f_{obj} = \lambda_2(t) \cdot \lambda_{2-tar.}^{-1} + \mathcal{A}(t) \cdot \mathcal{A}_{tar.}^{-1} \tag{19}$$

We remark that this list is clearly non-exhaustive: one could, for example introduce many more sophisticated scalarizing functions, such as one that evaluates as a step function for λ_2 and linearly (or quadratically) for $\mathcal{A}$.

All simulations started from the same initial pose, involved eight robots, and used hyper-parameters $O_p = 50$ (the frequency of the optimization) and $G_p = 400$ (the size of the search space). These values are meant to closely resemble the experimental setup ($O_p = 50$, $G_p = 250$) without risking G_p being too small to find interesting solutions (this is, nonetheless, proven not to be the case in Section 7). In Figure 8, we report the evolution of all relevant metrics—i.e., $f_{obj}(t)$, $\lambda_2(t)$, and $\mathcal{A}(t)$.

These results show that the scalarizing function in Equation 18 leads to larger values of area coverage $\mathcal{A}$ but unfairly penalize algebraic connectivty λ_2. This is motivated by the fact that, in our scenario, the domain of performance metric $\mathcal{A}(t)$ is typically larger than the domain of $\lambda_2(t)$. The results achieved with the scalarizing function in Equation 19 are comparable to those obtained when using the one in Equation 17. Equation 19, however, entails an additional

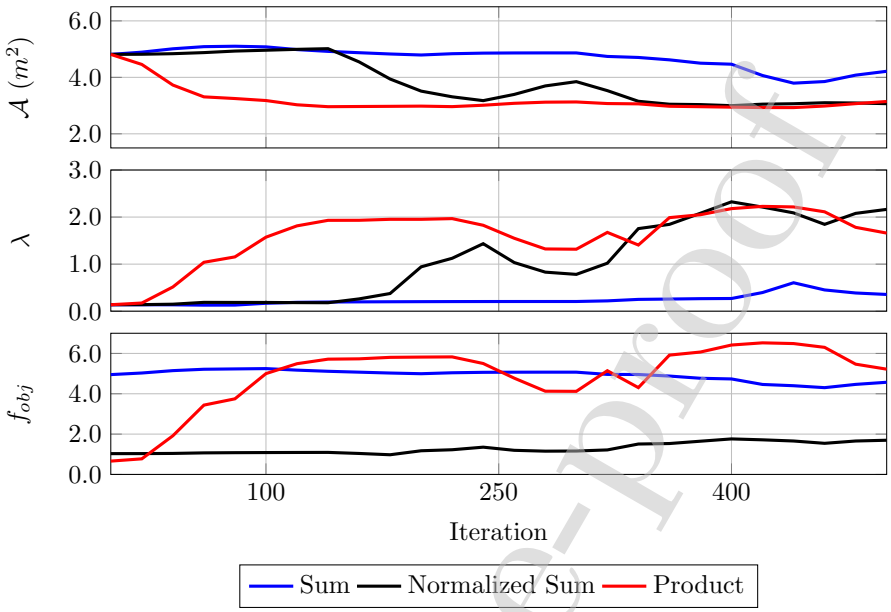


Figure 8: Evolution the performance metrics $\mathcal{A}$, λ and the scalarizing function f_{obj}, when optimization (i.e., a random search with $O_p = 50$, $G_p = 250$) is led by each of the three different implementation proposed in Appendix A. These results document the different performance of f_{obj} (and the two optimization objective λ and $\mathcal{A}$) when using different scalarizing functions (Equations (17)-(19)) and support our choice of using (17) in the rest of the article.

layer of complexity as it requires to run preliminary experiments to estimate the values of $\lambda_{2-tar.}$ and $\mathcal{A}_{tar.}$.

As our final goal is the evaluate the performance of autonomous, online optimization in the presence of faults, we opted to use Equation 17 as our preferred scalarizing function—following Dijkstra's opinion that complexity can pose a risk to reliability [52].

B. Fault injection

In this section, we outline the models and procedures that we used to inject faults within our simulations and experimental setup. Faults are meant to demonstrate manufacturing imperfections and other non-idealities afflicting the

physical world [53]. We use fault-injection to offer a more difficult challenge to the proposed control and optimization methodology.

In [53], faults are bipartite into two classes with respect to time duration: permanent and transient faults. Permanent faults perpetually affect a system since the time of their first occurrence. Transient faults can present themselves and then and disappear over time. In reliability engineering, probability distributions are typically used to model the initial time and the arrival times of permanent and transient faults, respectively [54].

Inspired by [39] and similarly to what we did in [11], we established protocols to inject two types of faults: (i) packet drop in the communication infrastructure—representative of transient/soft errors—and (ii) failures in the robotic hardware—representative of permanent/hard faults. The two following subsections detail these protocols.

B.1. Unreliable communication

Unreliable communication is implemented as the casual loss of certain packets/messages sent from one robot to another. Simulations and experiments with this sort of fault-injection replicate scenarios in which the robots' performance is distressed by faulty radios and/or environmental conditions (e.g., the presence of elevated electromagnetic interference).

We model the drops of messages as independent phenomena happening on each communication link, at a given rate. The likelihood of a message being dropped is described by a Bernoulli trial with probability mass function pmf_{Bern}:

$$pmf_{Bern}(sent, p) = \begin{cases} 1-p & \text{if } sent = \top \\ p & \text{if } sent = \bot \end{cases} \tag{20}$$

Table 4 reports the values of p in different phases of simulations lasting 500 iterations while Table 5 reports the values of p in different phases of experiments lasting ∼40'. To practically implement this model, we modified the software layer used to emulate point-to-point communication, i.e., `blabbermouth`.

it:	0–100	100–200	200–300	300–400	>400
p:	0.0	0.2	0.4	0.6	0.8

Table 4: Values of the packet drop rate over the development of each simulation

t:	0"–320"	320"–640"	640"–960"	960"–128'0'	>1280"
p:	0.0	0.2	0.4	0.6	0.8

Table 5: Values of the packet drop rate over the development of each experiment

B.2. Faulty robotic hardware

Robotic hardware failures intend to reproduce what would happen after the sudden disappearance of a drone flying within a swarm. In our fault-injection protocol, robots' failures happen independently and according to their mean-time-to-failure (MTTF). A robot's lifetime can be modeled using a probability distribution [55]. In our simulations/experiments, as we did in [11], we use an exponential cumulative distribution function CDF_{exp} is:

$$CDF_{exp}(t, \beta) = 1 - e^{-\frac{1}{\beta}t} \tag{21}$$

Hence, the MTTF equals the expected value: $E[X] = \beta$. In practice, the injection of robotic failures was implemented as follow: An initial grace period is granted for all robots. After the grace period ends, each robot's lifetime is regulated by an independent exponential distribution with MTTF of 300 iterations for simulations and ∼16' for experiments (60% of the simulation/experiment duration).

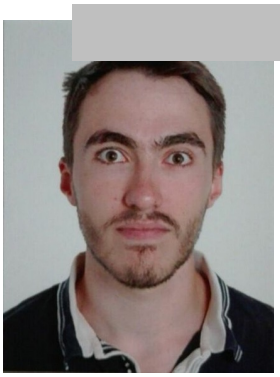
The occurrences of the failures—kept unaltered through all the experiments for each initial configuration—are summarized in Table 6 for simulations and in Table 7 for experiments. After a hard failure, a robot stops moving and communicating, in the simulation case, or it is physically removed from the arena in the experimental case.

Robot id:	2	3	4	0	7
Failure iteration:	232	247	322	375	397

Table 6: Robots' identifiers and failure iterations for simulations

Robot id:	3	6	2	1	9
Failure time:	7'14"	8'20"	9'45"	15'26"	18'29"

Table 7: Robots' identifiers and failure times for experiments



Marco Minelli is currently a M.Sc. student in mechatronic engineering at the University of Modena and Reggio Emilia (Italy) from which he received his B.Sc. degrees in mechatronic engineering in 2016.
His research interests include human-robot interaction, tele-robotics and multi-robot systems.

Jacopo holds a Ph.D. degree in computer engineering from Polytechnique Montréal (Montréal, QC, Canada). He received the M.Sc. degree in computer science from the University of Illinois at Chicago (Chicago, IL) in 2012, the Laurea Triennale degree in computer engineering from Politecnico di Milano (Milan, Italy) in 2009, and the Laurea Specialistica degree in computer engineering again from Politecnico di Milano (Milan, Italy) in 2011. In 2015, He was a visiting researcher at the National Institute of Informatics (Tokyo, Japan) and he attended the International Space University's Summer Study Program hosted by Ohio University (Athens, OH). In 2017, he served as a teaching associate for ISU at the Cork Institute of Technology (Cork, Ireland). Jacopo currently conducts post-doctoral research between Polytechnique Montréal and the European Astronaut Centre (Köln, Germany) in the context of ESA's Networking/Partnering Initiative. Jacopo's research interests include swarm robotics, human-robot interaction, machine learning, artificial intelligence, real-time and embedded systems.

Marcel Kaufmann is currently a Ph.D. student in Computer Engineering with the "Making Innovative Space Technologies" Laboratory at Polytechnique Montreal in Canada focusing on multi-robot systems, swarm technologies and human-robot interaction. He holds a B.Sc. and an M.Sc. degree in Photonics and Computer Vision from the University of Applied Sciences Darmstadt in Germany, from which he graduated in 2016. He then worked as a Scientific Software Engineer for the Dutch company Science [&] Technology on space, science and defense research projects. During the summer of 2017, he attended the International Space University's Space Studies Program hosted by the Cork Institute of Technology in Ireland.

Cinara Ghedini is a research associate at the company Energias. She received her M.Sc. in Computer Science in 2000 and her D.Sc. in Engineering and Computer Science in 2012. Between 2012 and 2014 she held a fellow research position at the Aeronautics Institute of Technology, Brazil. In 2015 she had been a Visiting Researcher at University of Modena and Reggio Emilia, Italy. Her main research interests involve complex networks, machine learning, failure tolerant systems and multi-agent robotics.

Giovanni Beltrame obtained his Ph. D. in Computer Engineering from Politecnico di Milano in 2006. He worked as microelectronics engineer at the European Space Agency on a number of projects, spanning from radiation-tolerant systems to computer-aided design. Since 2010 he is Professor at Polytechnique Montreal with the Computer and Software Engineering Department, where he directs the MIST Lab. His research interests include modelling and design of embedded systems, artificial intelligence, and robotics. He was awarded more than 15 grants by government agencies and industry, and has published more than 80 papers in international journals and conferences.

Giovanni Beltrame received the MSc degree in Electrical Engineering and Computer Science from the University of Illinois, Chicago, in 2001, the Laurea degree in Computer Engineering from Politecnico di Milano, Italy, in

2002, t
and a Ph.D. in Computer Engineering from Politecnico di Milano, in 2006. After his PhD he worked as microelectronics engineer at the European Space Agency and ESA Research Fellow on a number of projects spanning from radiation-tolerant systems to computer-aided design. In 2010 he moved to Montreal, Canada where he is currently Associate Professor in the Computer and Software Engineering Department of Polytechnique Montreal and director of the MIST Laboratory. He is principal investigator of more than 15 robotics and aerospace projects funded by government agencies and industry, and involved in the organization of several international conferences (e.g. ICRA, DATE, and others). His research interests include modeling and design of embedded systems, artificial intelligence, and robotics.

Lorenzo Sabattini received the B.Sc. and M.Sc. degrees in mechatronic engineering from University of Modena and Reggio Emilia, Modena, Italy, in 2005 and 2007, respectively, and the Ph.D. degree in control systems and operational research from University of Bologna, Bologna, Italy, in 2012. In 2010, he was a Visiting Researcher with University of Maryland, College Park, MD, USA. Since 2012, he has been an Assistant Professor in the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia. His main research interests include multirobot systems, decentralized estimation and control control, and mobile robotics.
Dr. Sabattini is one of the Founding Co-chairs of the IEEE Robotics and Automation Society Technical Committee on Multi-Robot Systems: he has served as the corresponding co-chair since its foundation, in 2014. He has been serving as an Associate Editor for IEEE Robotics and Automation Letters since 2015 and for IEEE Robotics and Automation Magazine since 2017.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: