An Integrated Topology Control Framework to Accelerate Consensus in Broadcast Wireless Sensor Networks

Massimo Vecchio, Gennaro Amendola and Pietro Ducange

***Abstract*—One of the primary constraints in the design and deployment of WSNs is energy, as sensor nodes are powered by batteries. In such networks, energy efficiency can be achieved by reducing the use of the onboard radios, for instance limiting packet transmissions. The broadcast nature of the wireless channel surely represents an advantage in this respect: each node has to send a single broadcast packet to simultaneously reach all its neighboring nodes, thus reducing the number of required transmissions. We present an integrated optimization framework leveraging on this advantage to improve the convergence speed of a distributed consensus algorithm, by means of topology design. We evaluate the effectiveness of the proposed framework in terms of overall energy savings and worst-case algorithmic complexity of the optimization task, on different classes of network topologies, and compare such results with those obtained by a pure greedy strategy recently proposed in the literature. We prove that our framework can slightly reduce the average nodes' energy cost with respect to its greedy antagonist, as well as reducing the computational overhead of the optimization task to a small fraction of the latter. These unique features make it suitable to tackle the problem also over large scenarios.**

***Index Terms*—Average consensus, algebraic connectivity, graph Laplacian, range assignment, topology control, wireless multicast advantage, greedy algorithms, look-ahead heuristics**

I. Introduction

Consensus is one of the most fundamental and ancient problems in distributed computing: in its most general meaning it consists in reaching an agreement on some value that depends on the state of all the agents involved in the computation. The *average consensus* is simply the agreement of the agents on their average state. Despite this plain definition and its age, consensus is back in vogue over the past few years, especially because of the hype around cryptocurrencies and related technology [1]. In this application domain, consensus is quite adopted (together with its derivatives like "consensus rules", "consensus formation algorithms", etc.) mainly to decide whether to commit a transaction to a distributed database, or to agree on someone's digital identity, etc. However, consensus is a thriving research topic also in other disciplines and contexts, extending from graph theory [2], systems science [3] and dynamic networks [4] to monitoring and control of industrial plants [5], and wireless communications [6], [7].

Regarding the latter and starting from the seminal work discussed in [8], a lot of research effort has been directed to the study of the average consensus problem in Wireless Sensor Networks (WSNs). The interest on this subject coming from the WSN community is confirmed by a fervent research activity of the last years: see, for instance, [9], [10], [11] and the references therein. Indeed, focusing on the WSN area, there exist several practical circumstances in which to solve the distributed consensus algorithm can become extremely useful: clock synchronization, intrusion detection, self-localization, sensor calibration and anomaly detection, only to mention few of them. The main reason for this attraction lies in the fact that distributed consensus algorithms only require iterative local information exchanges among the neighboring nodes and the computation of weighted sums at each node. This feature is extremely attractive when the goal is to aggregate data and information coming from several sources (*e.g.*, sensors) deployed in harsh, possibly large, areas. Recall, in the end, that this is one the most common assumptions of the Internet of Things in its practical declination [12]. Indeed, as stated in [5], WSNs can still play a key-role in situation awareness, namely in the mechanism of collecting, aggregating and mining knowledge from data for its posterior usage, which has a crucial importance when WSNs are deployed for monitoring and controlling, for instance in the context of living environments and/or industrial plants.

However, one important issue regarding distributed average consensus algorithms in WSNs is its *convergence speed*. Indeed, reducing consensus convergence time results in fewer transmissions, hence it has direct impact on nodes' battery lifetime. State-of-the-art approaches to accelerate convergence speed can be classified in two groups, namely:

1) *fixed network topology*: in this case, the only viable option is to optimize the edge weights, in order to minimize convergence time (see [13], [14], [15]);
2) *dynamic network topology*: in this case, the underlying network graph can somehow be altered. This is being reflected in additional flexibility, since optimization can be performed not only on the edge weights, but also on the graph itself [5], [16], [17], [18], [19], [20], [21]. We will refer to this approach as *topology optimization*.

Generally speaking, topology optimization is a very difficult combinatorial problem [22]; it follows that sub-optimal optimization approaches should be adopted, as done in the recent literature. For instance, in [16] the convergence properties of different topology classes are theoretically analyzed on average, given the number of nodes of a general network.

M. Vecchio is with the OpenIoT research unit, FBK CREATE-NET, Italy
G. Amendola and P. Ducange are with the SMARTEST Research Centre, eCampus University, Italy.

Moreover, authors of [18] start from a given topology and prove that removing links can be beneficial in terms of convergence speed. This seminal approach was then refined in [19], in order to judiciously remove and add links with the goal of speeding up the consensus algorithm, without increasing the overall network energy consumption. Finally, when dealing with WSNs with static nodes, topology control can be achieved by varying the nodes' transmission ranges (*e.g.*, by varying their radios' transmit power), as considered in [5], [17], [20], [21]. To the best of our knowledge, except [21], all of the previous approaches to topology optimization for improving consensus speed implicitly assume that the underlying network is *unicast*, so that links can be controlled independently. However, in WSNs there exists the possibility to exploit the broadcast nature of the wireless channel. More specifically, at each iteration, each node of the network broadcasts its state, while all of its neighbors simultaneously listen, and in this way the number of required transmissions is reduced. This is the reason why this concept is also known as Wireless Multicast Advantage (WMA) [23]. Unfortunately, when leveraging on WMA, the variation of the transmission range of a given node of the network affects the links to all of its neighbors, so that these cannot be independently controlled. This motivates the quest for specific topology optimization strategies that take these additional constraints into account.

The basic idea of this paper is to start from the only topology optimization technique that explicitly relies on the WMA to speed up the distributed consensus of the underlying network, namely, the *"full greedy (sparsification) strategy"* proposed in [21]. This technique is adopted as a reference strategy, both to derive alternative and more efficient techniques and to have a baseline for quantitative comparisons. More in detail, focusing on average consensus algorithms of the constant-weight type [24], our goal is to optimize each node's transmission range of a WSN that only relies on broadcast transmissions. Probably the first issue arising in such context is the fact that, if the transmission ranges of nodes are different, and assuming reciprocal channels, it may well happen that one node i is out of the coverage range of another node j, whereas node j can listen to node i's transmissions. In other words, the underlying graph becomes *directed*. Conditions for reaching average consensus with *directed topologies* are more demanding than those for the undirected case, since such networks usually require some sort of graph balancing [8], which is difficult to enforce in practice. To sidestep this problem and deal only with *undirected topologies*, we adopt the same strategy used in [21] where, at application layer, each node of the network simply drops those broadcast packets received from nodes not included in its own neighboring list.

More in detail, we propose a modular simulation and testing environment (*i.e.*, an *integrated framework*) able to facilitate the design and the assessment of various topology-control optimization algorithms, purposely forged to accelerate distributed consensus in broadcast WSNs. To this aim, Sec. II gives a preliminary introduction of the adopted system model, together with a formal introduction of the optimization problem to solve and some formal definitions and assumptions. Then, starting from the reference strategy proposed in [21] and adopting a top-down design, in Sec. III we identify and describe the three main functional blocks that, wired together, constitute the proposed integrated optimization framework. Moreover, to highlight the flexibility of such framework, in Sec. IV we substantially modify each functional block that actually constitute the realization of the baseline technique, so as to forge a new family of *quasi-greedy techniques* able to more efficiently tackle the original problem. Through the extensive simulation campaign described in Sec. V, we isolate a very specific member of the proposed family and for this technique we verify that, besides producing consistently better optimized solutions with respect to the baseline technique, it consumes only a small fraction of the computational power required to the baseline counterpart. These unique features make the selected technique particularly suitable when performing topology-control in WSN scenarios composed by hundreds of nodes and a single exact quality assessment of a network topology may become computationally very intensive, as it basically requires the inversion of an $n \times n$ matrix[1]. In these circumstances, it makes sense to:

1) reduce as much as possible the overall number of exact quality assessments performed during the execution of the topology optimization algorithm; and
2) reduce the computational time required to assess the quality of a single network topology.

The selected technique, being a direct realization of the proposed integrated optimization framework, is able to produce such better results with respect to the baseline counterpart because of a combination of *(i)* a more effective search strategy with respect to a standard greedy method, and *(ii)* a faster approximated quality assessment of a network topology. More in detail, we show that on large network scenarios the selected solution is able to save up to around 17% of the energy needed for the nodes to reach consensus, w.r.t. the greedy counterpart and at reduced computational cost, since it only requires the 2% of the total number of eigenvalue decompositions required by its antagonist. Concluding, as discussed in Sec. VI, the proposed optimization framework is able to accommodate more sophisticated and effective strategies with respect to pure greedy local search methods.

II. Preliminaries

This section introduces the main concepts and assumptions necessary to formally devise the proposed optimization framework. More in detail, Sec. II-A describes the adopted system model, Sec. II-B formally introduces the distributed consensus problem, while Sec. II-C details the model adopted in our framework to estimate the energy cost required to a WSN to reach the average consensus, in terms of communication cost and based on the actual nodes' transmission ranges.

A. Graph model

Consider a set V of uniformly and randomly deployed nodes with indexes $i \in \{1, \dots, n\}$. Let d_{ij} be the distance between nodes i and j, and let $\mathcal{R} = \{r_i \in [0, r_{\max}],\, i = 1, \dots, n\}$ be a

[1] n represents the total number of nodes of the network.

set of symmetric connectivity radii (transmission ranges, in the following), where $r_{\max} > 0$ denotes the maximum allowable range. In other words, $\mathcal{R}$ represents a given network configuration, in terms of transmission ranges of the comprising nodes. Then, we express the set of edges $E \subseteq V \times V$ of the graph $G = \{V, E\}$ in terms of $\mathcal{R}$, as

$$(i,j) \in E \quad \Leftrightarrow \quad d_{ij} \leq \min\{r_i, r_j\}. \tag{1}$$

Indeed, in this way the resulting graph is *undirected* by construction. At this point, it is worth noticing that, if $r_i = r \; \forall i$, then we recover the standard Random Geometric Graph (RGG) model [25]. However, with the aim of gaining flexibility during the optimization phase, we allow for different transmission ranges at different nodes. Obviously, this complicates the underlying range assignment (RA) problem that becomes NP-hard, as discussed in [26].

Once defined G, we can express the neighborhood of the generic node i in terms of the network's edges, as

$$\mathcal{N}_i = \{j \, : \, (i,j) \in E\}. \tag{2}$$

Finally, we define the elements of the $n \times n$ Laplacian matrix $\boldsymbol{L}$ of G, as

$$\boldsymbol{L}_{ij} = \begin{cases} |\mathcal{N}_i| & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } j \in \mathcal{N}_i \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

By construction $\boldsymbol{L}$ is symmetric. Let $\lambda_1(\boldsymbol{L}) \leq \lambda_2(\boldsymbol{L}) \leq \ldots \leq \lambda_n(\boldsymbol{L})$ denote its eigenvalues. Since all rows of $\boldsymbol{L}$ have zero sum, it follows from the Geršgorin theorem that $\boldsymbol{L}$ is positive semidefinite; in fact, one has $\lambda_1(\boldsymbol{L}) = 0$ with corresponding eigenvector the $n \times 1$ all-ones vector $\mathbf{1}$, since $\boldsymbol{L}\mathbf{1} = \mathbf{0}$. Moreover, $\lambda_2(\boldsymbol{L}) > 0$ if and only if G is connected [27]. The second smallest eigenvalue $\lambda_2(\boldsymbol{L})$ is known as the *algebraic connectivity* of the graph.

B. Average consensus algorithm

Let $\boldsymbol{x}(0) = [x_1(0) \; \ldots \; x_n(0)]^T \in \mathbb{R}^n$ denote the vector comprising the initial measurements at the nodes. The goal of the average consensus algorithm is to have all nodes compute the average of these values, $\bar{x} = \frac{1}{n}\mathbf{1}^T\boldsymbol{x}(0)$ iteratively and in a distributed fashion. Notice that, with the adjective *distributed* we mean that the generic node i of the network only communicates with nodes within its neighborhood $\mathcal{N}_i$. Moreover, we focus on constant-weight type distributed linear iterations [24], in which the state of node i is iteratively updated as

$$x_i(k+1) = x_i(k) + \alpha \sum_{j \in \mathcal{N}_i} [x_j(k) - x_i(k)], \tag{4}$$

where $\alpha > 0$ is a step-size. Letting $\boldsymbol{x}(k) = [x_1(k) \; \ldots \; x_n(k)]^T$ be the state vector at iteration k, we can write (4) as $\boldsymbol{x}(k+1) = (\boldsymbol{I} - \alpha\boldsymbol{L})\boldsymbol{x}(k)$. The goal is to have the states at all nodes converge to $\bar{x}$, given any initial condition $\boldsymbol{x}(0)$, *i.e.*:

$$\lim_{k \to \infty} (\boldsymbol{I} - \alpha\boldsymbol{L})^k \boldsymbol{x}(0) = \bar{x}\mathbf{1} = \frac{1}{n}\mathbf{1}\mathbf{1}^T\boldsymbol{x}(0). \tag{5}$$

It is well known that (5) holds if and only if $|1 - \alpha\lambda_i(\boldsymbol{L})| < 1$ for $2 \leq i \leq n$ [13] *i.e.*, if and only if $0 < \alpha < 2/\lambda_2(\boldsymbol{L})$. As in [13], we can define the *asymptotic convergence factor* as

$$\rho = \sup_{\boldsymbol{x}(0) \neq \bar{x}\mathbf{1}} \lim_{k \to \infty} \left(\frac{||\boldsymbol{x}(k) - \bar{x}\mathbf{1}||_2}{||\boldsymbol{x}(0) - \bar{x}\mathbf{1}||_2} \right)^{\frac{1}{k}} \tag{6}$$

and its corresponding *convergence time* as

$$\tau = \frac{s}{\log(1/\rho)}. \tag{7}$$

Specifically, τ is indicative of the number of iterations required for the error norm $||\boldsymbol{x}(k) - \bar{x}\mathbf{1}||_2$ to decrease by a factor of e^{-s}. Without loss of generality, in the following we will set $s = 7$ in (7), which amounts to a decrease of the error norm to 0.1% of its initial value.

Then, as shown in [13], we have

$$\rho = \max_{2 \leq i \leq n} |1 - \alpha\lambda_i(\boldsymbol{L})|, \tag{8}$$

therefore the optimum step-size α minimizing (8) is

$$\alpha_\star = \frac{2}{\lambda_n(\boldsymbol{L}) + \lambda_2(\boldsymbol{L})}. \tag{9}$$

Similarly, we can define the corresponding minimum value of ρ as

$$\rho_\star = \frac{1 - \gamma(\boldsymbol{L})}{1 + \gamma(\boldsymbol{L})}, \tag{10}$$

where the ratio

$$\gamma(\boldsymbol{L}) = \lambda_2(\boldsymbol{L})/\lambda_n(\boldsymbol{L}) \tag{11}$$

represents the objective function to maximize.

C. Consensus cost

First of all, it is worth to reveal in advance that the framework we are going to introduce in the next section behaves as a *direct search* optimization strategy [28], in the sense that the quality assessment of a solution of the problem is computed without the need of building any model for the objective function. This means that no assumptions are necessary on the latter continuity or derivability; in other words, the quality of a feasible solution is explicitly computed as a cost value that only depends on the actual solution's configuration and other system-wide parameters [29]. In this way, in our framework, complete freedom is left to the algorithm designer when defining the so-called *assess* block (see Sec. III-B).

Said that, the final goal of the optimization framework proposed in this paper is to modify a given broadcast WSN topology in a way that the convergence speed of the distributed consensus algorithm executed by the nodes comprising such network is improved. At the same time, it is understood that the ultimate cost we are interested in undermining is the *total energy* E spent by the network to reach such a consensus. Thus, in the following, instead of maximizing (11) we prefer optimizing the energy cost E_c, which is directly proportional to E and expressible in terms of the actual network configuration $\mathcal{R}$ as:

$$E \propto E_c(\mathcal{R}) = \frac{\tau}{n} \sum_{i=1}^{n} r_i^2. \tag{12}$$

In order to explain the last relationship, we notice that the energy cost E_c can be approximated by the sum of the *transmission* energy cost Etx and the *receiving* energy cost Erx necessary, on overage, to the nodes of the network to reach a consensus.

a) Transmission energy cost Etx: it is proportional to the average $\tau \cdot \mathrm{avg}\left(\mathrm{pow}_{\mathrm{tx}}\right)$ of the transmission powers. As done in [21], we assume that $\mathrm{pow}_{\mathrm{tx}}$ is proportional to r_i^{β}, where β represents the path-loss exponent and it set to 2, getting

$$Etx \propto \tau \cdot \frac{1}{N} \sum_{i=1}^{n} r_i^2. \tag{13}$$

b) Receiving energy cost Erx: it is proportional to the average $\tau \cdot \mathrm{avg}\left(\mathrm{pow}_{\mathrm{rx}}\right)$ of the receiving powers, which are proportional to the number of nodes within the transmission range of each node (*i.e.*, the number of *actual neighbors* $\overline{\mathcal{N}}_i$). Therefore, we have:

$$Erx \propto \tau \cdot \frac{1}{N} \sum_{i=1}^{n} \overline{\mathcal{N}}_i. \tag{14}$$

Since the nodes of the topologies we are taking into account are randomly and uniformly distributed in the unity square, $\overline{\mathcal{N}}_i$ is proportional to the area of the region in which such neighbors are, which is contained in the circle with radius r_i. Therefore, we have:

$$Erx \propto \tau \cdot \frac{1}{N} \sum_{i=1}^{n} r_i^2. \tag{15}$$

More precisely, since in our framework we allow different radii for the nodes, the whole circle is not needed and as a matter of fact the last expression is an upper bound for Erx.

Since Etx and Erx are proportional to (or proportionally bounded from above by) $\tau \cdot \frac{1}{N} \sum_i r_i^2$, also E_c is and, in the following sections, whenever we mention "*costs*", we will be implicitly referring to (12).

III. Architecture of the Proposed Optimization Framework

This paper takes roots in the seminal work described in [21], where the authors proposed some simple greedy strategies able to optimize the transmission ranges of a broadcast WSN, so as to accelerate the distributed average consensus algorithm running at sensor nodes. Starting from there, one of the goals of this paper is to provide a more flexible simulation-based implementation and testing environment able to generalize the very specific behaviour of the techniques proposed in [21] and, at the same time, to ease the forge and the assessment of new topology control techniques. Usually, in computer science terminology, such an environment is termed *framework*: a software platform modeling a specific relevant domain, where software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software [30]. In the remaining of this section, by adopting a classical top-down design, we will describe the main functional blocks that, wired together, constitute the proposed framework.

To begin with, let us consider the most generic *local search* strategy: a well-known direct search method that at each iteration moves from one solution to another in the space of candidate solutions (also known as *search space*). More specifically:

1) a move is performed by applying local changes to the current solution;
2) the quality of each move (*i.e.*, a candidate solution) is directly assessed by computing its fitness value according to a given objective function; and
3) among the assessed moves, the most advantageous one is selected for the next iteration of the search.

The previous actions are repeated until a stop criterion is verified (*e.g.*, maximum number of solution evaluations, elapsed time, etc.) and, at the end of the search procedure, the best solution found is returned [29]. Fig. 1 depicts the diagram of such a generic local search-based optimization framework, as composed by three main blocks, each of which implementing one of the three main actions of the search method, namely: *move*, *assess* and *select*. As we will see in the following, the optimization framework proposed in this paper belongs to this classical optimization scheme.

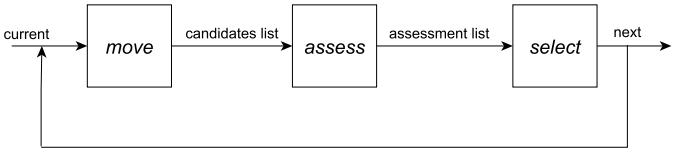


Figure 1. The block diagram of a general local search-based optimization scheme which the proposed framework belongs to.

A. *The* move *block*

Let us recall the main intuition behind the greedy strategies proposed in [21]: there, the problem of accelerating the average consensus algorithm was firstly mapped onto a more general range assignment (RA) broadcast problem. Then, starting from a maximally connected topology, all the proposed strategies successively removed one link at each iteration step (in this respect, they were all "sparsification" techniques). The selection of the link to remove fell into the most advantageous one (in this respect, in a purely greedy fashion) at the given iteration and in terms of a given metric (*e.g.*, convergence time, energy cost, etc.) among a purposely formed *list of candidate links* ("list of candidates", $Clist$ in the following). Finally, the way such lists were populated at each iteration derived the different greedy sparsification strategies. For instance, the best greedy strategy proposed in [21], namely the "full greedy strategy", simply populated $Clist$ at each iteration step by considering those links connecting each node of the network to its corresponding *farthest* neighboring node.

To generalize this intuition, we define a functional block able to output a list of candidate solutions, starting from a given graph representation (*e.g.*, as the corresponding symmetric Laplacian matrix $\boldsymbol{L}$) and according to a consistent set of rules. We term this functional block *topology-control policy-maker* and, by slight abuse of dynamic programming nomenclature, we say that the input Laplacian matrix represents the

current state, while the output is a list of *possible actions* [31]. As a proof of this concept, Algorithm 1 depicts a possible translation into a topology-control policy-maker block of the way the full greedy strategy described in [21] populated the list of candidates at each iteration t[2].

Algorithm 1. The translation of the specific strategy adopted by the full greedy strategy of [21] into a topology-control policy-maker program.

```
1: procedure FARTHESTNODEPOLICYMAKER
Input: D, L_t
Output: Clist_t
2:     Clist_t ← emptyList
3:     for i ← 1 to n do
4:         j ← FARTHESTNODE(i, L_t, D)
5:         Clist_t.PUSH(GETLINK(i, j))
6:     end for
7:     return Clist_t
8: end procedure
```

Notice that Algorithm 1 uses an internal utility function FARTHESTNODE to isolate the index j of the farthest neighboring node of the given node i. This is due to the intrinsic notion of "farthest neighbor" that cannot avoid considering, besides the Laplacian matrix of the underlying undirected graph, the actual network topology (*e.g.*, the immutable x-y coordinates of each node of the topology or, equivalently, the global $n \times n$ distance matrix, $\boldsymbol{D}$ in Algorithm 1). Due to severe space limitation, the implementation of this utility function is omitted.

B. *The* assess *block*

In the previous section, we highlighted the crucial role played within our framework by the policy-maker block when designing effective, but computationally affordable, topology-control optimization algorithms. To step forward in the description of the proposed framework, it is worth looking at the action of picking the best candidate out of $Clist$ ("picking action", in the following) as composed by two elementary sub-actions, namely:

1) the *assessment* of the quality of each candidate solution, according to a given metric; and
2) the *selection* of the best assessed candidate solution, according to a given decision strategy.

Therefore, in the proposed framework, the picking action is further decomposed into two separate functional blocks. This approach provides two main advantages, detailed next. The first advantage is architectural: to fix the ideas, it is worth recalling that all the strategies proposed in [21] evaluated the quality of a candidate solution by estimating the convergence time of a consensus algorithm running on the nodes (or, equivalently, the average energy cost associated to this execution) of the underlying network topology. There, to perform such estimation, the heaviest operation (in terms of computational power) was the eigenvalue decomposition of the Laplacian matrix representing the actual network topology. More formally, all the strategies proposed in [21] assessed the quality of the candidate solutions by solving (7) (or, equivalently, (12)). Obviously, the latter is only a way to assess the quality of a candidate solution, while alternative assessment functions can also be defined. Thus, to have a dedicated functional block responsible of the quality assessment of a topology adds flexibility to the proposed framework. The second advantage is keener and leverages on the intuition of relying on faster quality assessment functions for the candidate solutions, even at the cost of obtaining less accurate cost estimations. Indeed, by speeding up the process of estimating each candidate solution contained in $Clist$, it would be possible to evaluate more candidate solutions per time unit. Hence, the possibility of designing and plugging computationally-faster assessment blocks delivers to our framework the non-obvious feature of accommodating topology-control strategies relying on lowly restrictive policy-makers implementations (*i.e.*, policy-maker blocks producing larger candidate lists at each iteration). Said differently, the proposed framework is able to trade-off some accuracy in the $Clist$ assessment with an extended overall exploration capability of the search space.

[2]In the following, we adopt the notation X_t to say that variable X depends on iteration t.

C. *The* select *block*

As introduced in the previous section, the select block is responsible of selecting the best assessed candidate out of $Clist$ and according to a given decision strategy. By adopting the same approach used to introduce the other blocks of the proposed framework, it is worth to recall how such primitive action was implemented by the techniques described in [21]. There, at each iteration of the search procedure, the candidate solution characterized by the minimum estimated convergence time (or, equivalently, the minimum average energy cost) was selected as the topology for the next iteration. In this sense, all the strategies proposed there were intrinsically greedy. However, as for the assessment block, this is not the only way a direct search algorithm has to implement the *select* move. On the contrary, since this specific RA problem does not exhibit neither an optimal sub-structure, nor a greedy-choice property [32] (otherwise the problem would have not been NP-hard, as proved in [22]), from a research point of view it is extremely interesting to explore alternative strategies to more effectively tackle it. Thus, with the goal of enabling such possibility, our framework abstracts the *select* action, providing such action as a dedicated functional block.

IV. A FAMILY OF QUASI-GREEDY APPROXIMATED TECHNIQUES

In the previous section, we thoroughly described the three main functional blocks that wired together compose the proposed optimization framework. Moreover, to accentuate its generalization capabilities, we showed how the original full greedy strategy proposed in [21] could be ported into such framework. Along the same lines, in this section, we will introduce some alternative implementations of the functional blocks that, combined together, constitute a new *family* of optimization techniques suitable for more efficiently tackling the original RA problem. We term this family *"quasi-greedy approximated techniques"*.

A. Move block: unconstrained link removal policy-maker

In Sec. III, we drew attention on the crucial role played by the move block when designing effective but computationally affordable topology-control algorithms. Using a more optimization-friendly terminology, we may say that such block is responsible of building the current solution's "neighborhood", in a Variable Neighborhood Search (VNC) fashion [33]. Generally, this means that the larger is the average neighborhood size, the wider would be the portion of the search space explored by the algorithm. Therefore, when no other options exist to balance the trade-off between the exploration capabilities and the computational overhead of a direct search technique, it is critical to adopt some wisely-designed move strategy. As per the discussion of the previous section, in this specific context, we know that this is not the case for our framework where, downstream of the policy-maker block, options to balance such trade-off do exist, even though the implementation details have not yet been disclosed.

Therefore, all the members of the family proposed in this section rely on a not-so-restrictive policy that is here described: the direct search starts considering, at iteration $t = 0$, a highly-connected topology where all nodes' transmission ranges are set to a common value (*i.e.*, $\mathcal{R}_{t=0} = \{r_i = r_{\max},\ \forall i\}$); from such network topology, the associated symmetric Laplacian matrix $\boldsymbol{L}_{t=0}$ is considered. Then, at each iteration t of the algorithm, the policy-maker block simply populates $Clist_t$ by pushing all the existent (undirected) links of $\boldsymbol{L}_t$. We term this policy *unconstrained link removal policy*, to emphasize that it is still a sparsification technique (as the farthest-node policy described in Algorithm 1 is), though there are no constraints for an existent link of the current network topology to be considered (as opposed to the farthest-node policy described in Algorithm 1).

For the sake of completeness, the translation of this policy into a program is depicted in Algorithm 2, while in Sec. IV-D we will formally prove that this policy represents a less restrictive version of the farthest-node policy described in Algorithm 1. Finally, regarding the enforcement of such policy at sensor nodes, we can assume, for instance, that each node has knowledge of the list of neighboring nodes belonging to the optimized network configuration. Hence, each receiving node simply drops those broadcast packets received by nodes not belonging to its own optimized neighboring list.

Algorithm 2. The *unconstrained link removal policy-maker* program.

```
1: procedure MOVE
Input: L_t
Output: Clist_t
2:     Clist_t ← emptyList
3:     for i ← 1 to n do
4:         for j ← i + 1 to n do
5:             if L_t(i, j) == −1 then
6:                 Clist_t.PUSH(GETLINK(i, j))
7:             end if
8:         end for
9:     end for
10:    return Clist_t
11: end procedure
```

B. Assess block: approximated quality assessment

In the context of the proposed family of topology-control techniques, the main requirement for a suitable quality assessment block is to be computationally fast. Recall, in fact, that the move block adopted by all the techniques of this family is quite loose, hence generating, at each iteration of the search, several candidate topologies that need to be assessed.

Before introducing the implementation details of the three quality assessment blocks of this family, it is important to recall that, from an algebraic point of view, if the edge (s, d) connecting nodes s and d is removed from graph G represented by $\boldsymbol{L}$, then the new Laplacian becomes

$$\widetilde{\boldsymbol{L}} = \boldsymbol{L} - \omega \boldsymbol{g}\boldsymbol{g}^T, \tag{16}$$

where $\boldsymbol{g} = \boldsymbol{e}_s - \boldsymbol{e}_d$, $\boldsymbol{e}_k$ is the k-th column of the $n \times n$ identity matrix and $\omega = 1$. For this reason, $\widetilde{\boldsymbol{L}}$ is a rank-1 perturbation of $\boldsymbol{L}$ [21]. Given that the goal is to maximize (11), to speed up such computation the proposed assessment blocks only *estimate* $\gamma(\widetilde{\boldsymbol{L}})$ without actually performing the eigenvalue decomposition of $\widetilde{\boldsymbol{L}}$. We start by considering (16), this time expliciting the dependency of $\widetilde{\boldsymbol{L}}$ on $\omega \in \mathbb{R}$, as

$$\widetilde{\boldsymbol{L}}(\omega) = \boldsymbol{L} - \omega \boldsymbol{g}\boldsymbol{g}^T. \tag{17}$$

Thus, $\widetilde{\boldsymbol{L}}$ is a function of ω, as indeed its eigenvalues (and eigenvectors), and we are interested in approximating the ratio of its second and n-th eigenvalue, when $\omega = 1$ (*i.e.*, the rank-1 perturbation after removing an edge of $\boldsymbol{L}$). To this aim, we use three classic approximation methods, namely the first and the second order Taylor and the log-linear approximations (1st-order, 2nd-order and log approximations, in the following).

In what follows, the i-th eigenvalue of $\widetilde{\boldsymbol{L}}(\omega)$ and its corresponding eigenvector will be denoted by $\lambda_i(\omega)$ and $\boldsymbol{v}_i(\omega)$, respectively. We can suppose that $\lambda_i(\omega)$ and $\boldsymbol{v}_i(\omega)$ are differentiable w.r.t. ω and, since $\widetilde{\boldsymbol{L}}(\omega)$ is symmetric, that the $\boldsymbol{v}_i(\omega)$'s are orthonormal. From [34] we have that, for $i = 1, 2, \ldots, n$, the first and and the second-order derivatives of λ_i w.r.t. ω are

$$\lambda_i'(0) = -\left(\boldsymbol{v}_i^T \boldsymbol{g}\right)^2, \tag{18}$$

$$\lambda_i''(0) = \sum_{\substack{j=2,3,\ldots,n \\ j \neq i}} 2\frac{\left(\boldsymbol{v}_i^T \boldsymbol{g}\right)^2 \left(\boldsymbol{v}_j^T \boldsymbol{g}\right)^2}{\lambda_i - \lambda_j}. \tag{19}$$

For our family of techniques, we consider the following three approximations of $\gamma(\widetilde{\boldsymbol{L}})$:

a) 1st-order approx: the Taylor approximation of the ratio up to first order

$$\gamma(\widetilde{\boldsymbol{L}}) \simeq \gamma(\boldsymbol{L}) + \gamma'(0); \tag{20}$$

b) 2nd-order approx: the Taylor approximation of the ratio up to second order

$$\gamma(\widetilde{\boldsymbol{L}}) \simeq \gamma(\boldsymbol{L}) + \gamma'(0) + \frac{1}{2}\gamma''(0); \tag{21}$$

c) log approx: the Taylor approximation of the second eigenvalue up to second order and the log-linear approximation of the n-th eigenvalue up to first order, obtained by means of a suitable change of variable

$$\lambda_2(\widetilde{\boldsymbol{L}}) \simeq \lambda_2(0) + \lambda_2'(0) + \frac{1}{2}\lambda_2''(0), \tag{22}$$

$$\lambda_n(\widetilde{\boldsymbol{L}}) \simeq \lambda_n(0) - \frac{\left(\lambda_n'(0)\right)^2}{\lambda_n''(0)} \left(\log\left(1 - \frac{\lambda_n'(0)}{\lambda_n''(0)}\right) + \log\left(-\frac{\lambda_n'(0)}{\lambda_n''(0)}\right)\right) \tag{23}$$

For the sake of completeness, Algorithm 3 depicts a possible implementation of the approximated quality assessment block: at each iteration t of the search, it takes as input the current network configuration $\boldsymbol{L}_t$ and the current list of candidates $Clist_t$ as provided by the policy-maker block and outputs the list of approximated quality estimations $estQ_t$, depending on the specific approximation method and passed as input argument AP_M.

Algorithm 3. The *approximated quality assessment* program.

procedure ASSESS
Input: $\boldsymbol{L}_t$, $Clist_t$, AP_M
Output: $estQ_t$
  $estQ_t \leftarrow emptyList$
  for $j \leftarrow 1$ **to** SIZE($Clist_t$) **do**
    $\tilde{\boldsymbol{L}}_t = \boldsymbol{L}_t - Clist_t[j]$
    $estQ_t[j] \leftarrow$ APPROXEVAL($\tilde{\boldsymbol{L}}_t, AP\_M$)
  end for
  return $estQ_t$
end procedure

C. Select block: quasi-greedy selection

In this section, we describe the implementation of the selection block adopted by the members of the proposed family. First of all, it is worth remarking that, in the economy of the framework, the role covered by this block is at least as important as the other two, since its implementation exploits some assumptions and outcomes coming from the upstream blocks. Thus, this block can be seen as the gluing component providing the whole family with the appearance of an integrated optimization framework. Overall, the main macro-effect of this block is to steer the search ability of the family towards not-purely-greedy directions, so we term it *"quasi-greedy selection block"*.

The block performs as following: at each iteration t, it sorts the elements of $estQ_t$ in ascending order. Then, it scans one by one such sorted elements, starting from the most promising estimated quality and until either verifying the effectiveness of a certain element, or reaching the end of the list. In some sense, this process resembles an *auction system*, in which the elements of $estQ_t$ represent the *bids*; before accepting one bid, the latter is firstly verified, according to some given constraints imposed by the equivalent of a *surveyor*. Upon the *verification* of a bid, the block selects it for the next iteration of the search, hence *correctly* closing the auction; otherwise the next bid is processed, and so on and so forth, until verifying (and selecting) a bid, or reaching the end of the sorted list. In the latter case, all bids were not verifiable and, with the proviso of storing the best actual cost found throughout the whole verification process, such bid is forcefully verified and selected. This because the algorithm must necessarily perform a selection at each iteration, eventually at the cost of hill-climbing: in this case we say that the auction is *not correctly* closed.

Generally speaking, complete freedom is left in the definition of the set of conditions that together form a surveyor. For instance, simple surveyor implementations may require bids to be within a certain interval of real costs, or simply better than the current cost, and so on and so forth. Nevertheless, it is worth noticing that the more demanding the surveyor is, the more difficult would be to correctly close an auction. This happens because the surveyor will have to perform more verifications, before actually selecting one bid and moving the algorithm to the next iteration step. Even worse, too-hard-to-meet conditions could cause whole auctions not to be correctly closed: this means that, at a given step, all bids are not verifiable and the "least bad" would be selected only afterward. The opposite is also true: too loose surveyor's conditions mean weakening the beneficial effects of the quasi-greedy heuristics, by instilling too much trust in the approximated assessment block.

From a mere coding perspective, a bid is verified by exactly assessing the average energy cost associated to the corresponding Laplacian perturbation. Because of this, at iteration t of the main optimization loop, the exact evaluation of a verified bid can be saved and re-used at iteration $t+1$; this means that at step $t+1$, the optimization engine has not to compute the eigenvalue decomposition of $\boldsymbol{L}_{t+1}$, as this computation was already performed at step t to verify the associated bid. This results particularly useful when, for instance, the condition imposed by the surveyor adopts the quality of the current Laplacian configuration as a reference value for verifying bids. Therefore, with slight oversimplification, the implementation of such quasi-greedy block is similar to that described in [35] and for this reason we term it *look-ahead heuristics*.

Regarding the actual definition of the rules realizing the surveyor of the proposed family, we conducted a wide offline simulation campaign, out of which it emerged that simple comparisons between the cost associated to the current network configuration $\boldsymbol{L}_t$ and those associated to $\tilde{\boldsymbol{L}}_t$ are sufficient. Therefore, all the techniques of the proposed family adopt the following single-rule-based surveyor:

$$\text{EXACTCOST}(\boldsymbol{L}_t) > \text{EXACTCOST}(\tilde{\boldsymbol{L}}_t), \tag{24}$$

where EXACTCOST(x) computes the energy cost (as detailed in (12)) associated to an input network configuration x, represented as a symmetric Laplacian matrix. In other words, the proposed surveyor "verifies a bid and correctly close the current auction" as soon as the Laplacian perturbation under verification ensures a down-hill (*i.e.*, a real cost reduction w.r.t. the current network configuration $\boldsymbol{L}_t$).

Although the surveyor summarized in (24) might seem too simplistic, it is worth highlighting that, during the search, it is not a rare event that of *not correctly* closing an auction. In terms of number of verifications, this translates into a surveyor exactly evaluating all of the Laplacian perturbations and selecting the "least bad" one only a-posteriori. Transposing this argument onto the energy cost curve to minimize, the optimization algorithm could not find a down-hill and

reluctantly had to select an up-hill. Thus, it makes sense to deterministically combat this pathological situation that ultimately only wastes computational power. In our framework this is realized by limiting the maximum number of bids to verify at each iteration. More practically, the look-ahead procedure is not applied exhaustively to all of the elements of $Clist_t$, but only to the $laSize < \text{SIZE}(Clist_t)$ most promising ones. It is clear that, to wisely set the *look-ahead size* (*i.e.*, to dimension $laSize$) is as crucial as establishing well-balanced surveyor's conditions. In the remaining of this paper, this concept will be made more evident by simulations while, for the moment, we simply consider the availability of the LIMITLASIZE() method for this purpose, to realize different members of the same optimization family.

For the sake of completeness, a possible implementation of this block (comprising the surveyor and the look-ahead blocks) is depicted in Algorithm 4. The program takes as input the current network configuration $\boldsymbol{L}_t$ and its associated energy cost c_t, together with the list of candidates $Clist_t$ (as produced by the policy-maker block) and their corresponding estimated qualities $estQ_t$ (as produced by the approximated assessment block). The output of this block is the selected Laplacian perturbation $\boldsymbol{L}_{t+1}$ and its associated cost c_{t+1}. Notice that the program relies on the $[sv, si] = \text{SORTASCENDING}(x)$ method that takes a sequence of unsorted values x as input, and outputs the arrays sv and si, where sv lists the sorted values and si contains the corresponding indexes of x. In this way, it is possible to directly access the s-th sorted element of x, by using the s-th index contained in si, as done, for instance, in line 7 of Algorithm 4.

Algorithm 4. The *quasi-greedy selection* program.

```
 1: procedure SELECT
Input: L_t, c_t, Clist_t, estQ_t
Output: L_{t+1}, c_{t+1}
 2:     [sB, sI] ← SORTASCENDING(estQ_t)
 3:     laSize ← LIMITLASIZE()
 4:     s ← 1
 5:     best ← +∞
 6:     for j ← 1 to laSize do                          //bids verification
 7:         cost[j] ← EXACTEVAL(L − Clist[sI[j]])
 8:         if cost[j] < best then
 9:             s ← j                                   //keeping the "least bad" index in s
10:             if cost[j] < c_t then                   //surveyor's condition of (24)
11:                 break                               //bid verified: auction correctly closed
12:             end if
13:         end if
14:     end for
15:     L_{t+1} ← L − Clist[sI[s]]
16:     c_{t+1} ← cost[s]
17:     return [L_{t+1}, c_{t+1}]
18: end procedure
```

D. Worst-case complexity analysis

At this point, it is clear that an algorithm designer has different ways to balance the trade-off between the overall search capability of a technique belonging to the proposed family and its computational overhead. However, as seen in [21], the most expensive operation that the generic algorithm executes at each iteration of the search is the eigenvalue decomposition (EVD) of the Laplacian matrix associated to a candidate topology. Thus, it is important to formally analyze the worst-case complexity of a given technique, so as to give upper bounds on the computational resources required by the algorithm, terms of *total number of EVDs* ($\mathcal{E}$, in the following).

Without loss of generality, we assume that a given algorithm is executed for M iterations; if e_w represents the number of EVDs that it executes at each iteration in the worst case, then we can derive an upper bound for $\mathcal{E}$ as:

$$\mathcal{E} \leq M \cdot e_w. \tag{25}$$

For instance, considering the full greedy strategy proposed in [21], at most n EVDs are necessary at each iteration of the search, so a possible upper bound $\mathcal{E}_{BASE}$ for such strategy could be:

$$\mathcal{E}_{BASE} \leq M \cdot n. \tag{26}$$

With respect to the proposed family of quasi-greedy techniques, because of the arguments of the previous section, a loose upper bound $\mathcal{E}_F$ could be:

$$\mathcal{E}_F \leq M \cdot laSize_{max}, \tag{27}$$

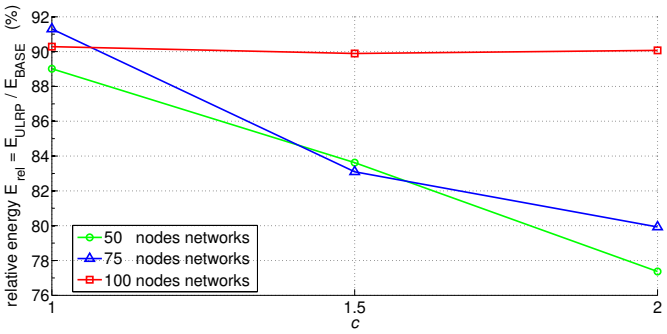
where $laSize_{max}$ represents the maximum size reached by the look-ahead, throughout the execution of a given algorithm of such family. Clearly, if such algorithm adopted a constant and fixed size $laSize$ for the look-ahead, then $M \cdot laSize$ would represent a tighter upper bound for $\mathcal{E}_F$.

V. Experimental results

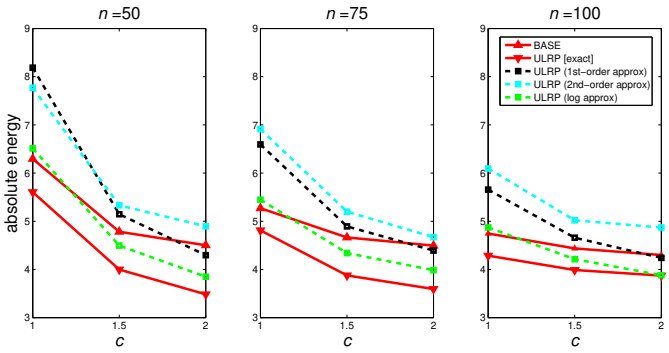
This section describes the simulation results obtained by different topology-control strategies on various synthetically-generated WSN scenarios. Specifically, we have generated several network topologies by randomly and uniformly placing n nodes in $U = [0, 1] \times [0, 1]$. Moreover, to further differentiate the network scenarios, we have varied the value of the initial maximum transmission range as a function of n, as $r_{\text{max}} = \sqrt{c\frac{\log n}{n}}$. Thus, after fixing n, the larger is the value of c, the denser would be the topology. For all the experiments described in this section, if not differently stated, we have varied $n \in \{50, 75, 100\}$, $c \in \{1, 1.5, 2\}$ and we have averaged the results over 30 deployments for each (n, c) pair. Regarding the topology-control strategies, when possible, we have used the results of the full greedy strategy proposed in [21] as baseline reference values. Regarding the number of iterations of any of the assessed direct search methods, if not differently stated, we have fixed $M = 500$.

A. First experiment

In Sec. IV-A, we revealed that all the different members of the proposed quasi-greedy optimization family adopt the same policy-maker block implemented by Algorithm 1. For this reason, it is of primary importance to assess the impact of this block's implementation on the overall performances of the techniques of the family. Moreover, as introduced in Sec. IV-B, different approximated assessment methods characterize different members of the family. Thus, it is also important to assess how different approximation methods affect the overall performances of the corresponding techniques. The objective of this experiment, divided into two phases, is to quantitatively measure such impacts.



(a) Phase one



(b) Phase two

Figure 2. First experiment: comparison, in terms of relative and absolute energies, between the farthest-node removal policy (BASE) and the unconstrained link removal policy (ULRP) in its (a) ideal and (b) approximated variants.

a) ideal policy maker block (phase one): in this phase, we simply compare the results obtained by the original full greedy strategy proposed in [21] (BASE, in the following) with those obtained by an *ideal* variant of the same strategy. Specifically, the latter adopts the unconstrained link removal policy implemented in Algorithm 1 (ULRP, in the following) instead of the original farthest-node policy of Algorithm 2. Fig. 2(a) depicts a comparison of such techniques, in terms of relative energy $E_{rel} = E_{ULRP}/E_{BASE}$. Since the relative cost is proportional to the actual average energy spent by the nodes of the network, the goal is to minimize the ratio. Hence, lower values of E_{rel} correspond to better payoffs for ULRP w.r.t. BASE. Observing Fig. 2(a), we say that ULRP consistently finds better solutions w.r.t. BASE; moreover, the benefits of an unconstrained link removal policy seem more prominent in denser scenarios. However, this variant of BASE is only ideal: recall, in fact, that the derived ULRP technique is computationally heavy, since the assessment and selection blocks are the original ones adopted by BASE. In other words, this version of ULRP relies on the unconstrained removal policy-maker, without exploiting any fast assessment block. This is reflected on the excessively large number of EVDs that such a technique has to compute at each iteration ($e_w \leq n^2$, so $\mathcal{E}_{ULRP} \leq M \cdot n^2$ is a reasonable upper bound of the total number of EVDs). Nevertheless, this phase of the experiment is only preparatory, as its goal is only to prove by simulation that a less restrictive link removal policy can be beneficial in terms of overall search capability of the family.

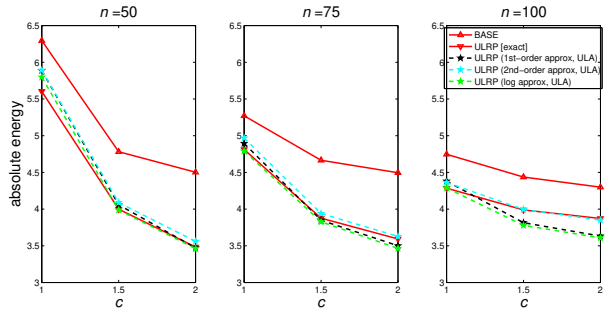
b) real policy maker blocks (phase two): in this phase, besides BASE and the ideal version of ULRP, we compare three more *realistic* variants against each other (see Fig. 2(b) where, to ease multiple comparisons, we depict the absolute energy costs of each technique). More in detail, the results of BASE and ideal ULRP (labeled as *exact*) are shown together with the results obtained by three approximated ULRP variants adopting the 1st-order, 2nd-order and log approximation blocks introduced in Sec. IV-B, respectively. We observe that only the log approximation block could approach the results of the exact counterpart, especially for very dense scenarios (*e.g.*, $n = 100$ and $c = 2$). On the other hand, it is worth noticing also that, except for too sparse scenarios (*e.g.*, $\forall n$ and $c = 1$), the log-approximated ULRP method has already better search capabilities w.r.t. the reference strategy proposed in [21].

B. Second experiment

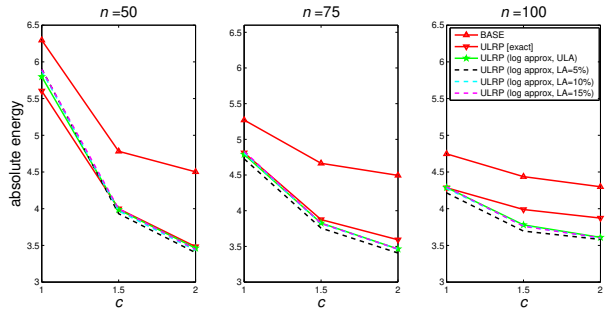
In Sec. IV-C, we introduced the possibility of realizing different members of the proposed quasi-greedy optimization family, by purposely changing the implementation of the LIMITLASIZE() method of Algorithm 4. With this experiment, we assess the impact of some specific realizations of such method on the overall performances of the derived techniques. This experiment is divided into three phases.

a) unbounded look-ahead block (phase three): in this phase, we enable also the select block described in Sec. IV-C into the three approximated variants of the previous phase. Specifically, for such members of the family and regarding the LIMITLASIZE() method, we simply leave unbounded the size of the look-ahead heuristics. This means that, at each iteration t, LIMITLASIZE() simply returns the size of $Clist_t$, thus admitting all the candidate topologies produced by the policy-maker block to the current auction. We term the members of the proposed family using such a select block *"unbounded look-ahead"*-based strategies, ULA in the following. Notice that $\mathcal{E}_{ULA} \leq M \cdot n^2$, though looser w.r.t. ULRP, is still a valid upper bound of the total number of EVDs. As expected, the strategy adopting the log approximation still obtains the best results. More interestingly, we observe that, except for the sparsest scenarios (*e.g.*, $n = 50$ and $c = 1$), the technique adopting the logarithmic approximation block followed by the ULA block is able to perform at least as good as its exact ULRP counterpart. Moreover, for sufficiently dense scenarios (*e.g.*, $\forall n$ and $c > 1$), the former is able to obtain better results w.r.t. the latter. This fact confirms that pure greedy algorithms cannot guarantee to reach optimal solutions of such a complex optimization problem. In other words, a pure greedy method like the exact variant of ULRP, cannot trace insurmountable (lower) bounds for our optimization framework, in terms of optimized topologies. This argument is further analyzed in the following phases.

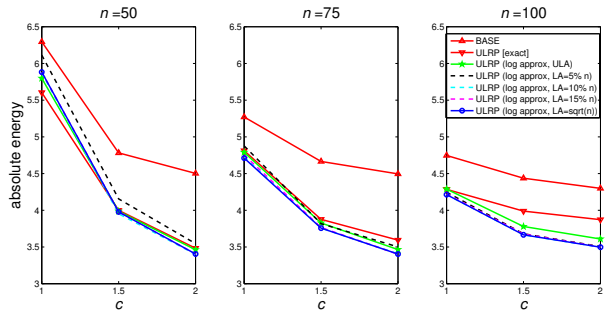
b) Dynamically-bounded look-ahead blocks (phase four): motivated by the encouraging results of the previous phase, in the following we aim at better measuring the impact of the look-ahead size on the global search capability of the



(a) Phase three



(b) Phase four



(c) Phase five

Figure 3. Second experiment: comparison, in terms of absolute energy, between the farthest-node removal policy (BASE) and the unconstrained link removal policy (ULRP) with (a) unbounded, (b) dynamically-bounded and (c) statically-bounded look-ahead blocks.

corresponding techniques. As briefly introduced in Sec. IV-C, the implementation of the LIMITLASIZE() method plays a crucial role in the economy of the proposed framework and family of optimization strategies, since reducing (*i.e.*, *bounding*) the look-ahead size is the way the techniques belonging to this family actually trade-off computational complexity and search power. In this phase, we reduce the size of the look-ahead to a fraction of its own current size (*i.e.*, *dynamically*), so a reasonable upper bound of the total number of EVDs can be the one summarized in (27). In the following, for the sake of visual clarity, we omit reporting the results obtained with the 1st and 2nd-order approximation blocks, focusing only on assessing the performances of the strategies adopting the logarithmic approximation block. Fig. 3(b) compares the curves of Fig. 3(a) with those obtained when reducing at each iteration t the look-ahead size to 5, 10 and 15 percent of the actual $Clist_t$ size[3]. Observing Fig. 3(b), we understand that limiting the look-ahead size may be beneficial not only to save computational power, but also to improve the effectiveness of the optimized solutions. Indeed, the technique limiting the look-ahead size to 5% is the best assessed one, while larger look-ahead sizes (*e.g.*, 10% and 15% of $Clist_t$ size) do not guarantee better results in terms of energy cost w.r.t. the unbounded counterpart.

c) Statically-bounded look-ahead blocks (phase five): in the previous phase we observed that for our family of optimization techniques, if we want to improve the effectiveness of the search while also keeping the computational complexity at bay, then we have to dynamically set the look-ahead size to quite small values w.r.t. the actual $Clist_t$ size. It follows that it makes sense to explore also simpler bounding strategies, that are eventually dependent on some global network parameters that remain constant throughout the optimization phase (*i.e.*, *statically*). Moreover, to statically bound the look-ahead size has the non negligible advantage of a-priori defining tighter upper bounds of the algorithmic complexity, in terms of number of EVDs. For this reason, in this phase, we decided to set the look-ahead size to 5, 10, 15 percent of the total nodes n and to $\sqrt{n}$, respectively (square roots and percentage values are rounded down). The results of this experiment are shown in Fig. 3(c), where we observe that the technique adopting a look-ahead size equal to $\sqrt{n}$ is able to reach equivalent results w.r.t. its siblings. For this reason, we select and keep it as the only representative of the whole family, for the next experiment.

C. Third experiment

The wide simulation campaign described in Sec. V-A and Sec. V-B ended up suggesting that a technique adopting:

- an unconstrained link removal policy, as *move* block;
- a logarithmic approximation block, as *assess* block; and
- a statically-bounded look-ahead size set to $\sqrt{n}$, as *select* block

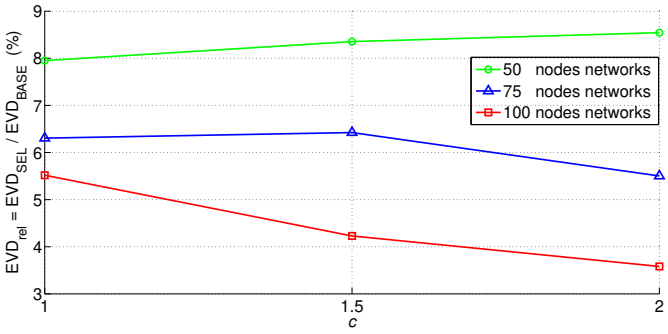
is quite representative of the whole family of optimization techniques proposed in this paper and implemented in our framework. Therefore, after re-labeling it as *"selected technique"* (SEL, in the following), in this experiment we aim at further assessing its effectiveness.

a) Comparing energy costs (phase six): first of all, it makes sense to compare the relative energy cost of the selected technique w.r.t. the baseline technique. This is done in Table 4(a), where we observe that SEL consistently finds better solutions w.r.t. BASE. Specifically, this table reports *relative energy gains*, which are expressed in percentage and computed as 100 minus the corresponding relative energy cost of SEL w.r.t. BASE. We observe that using the final optimized topologies searched by SEL ensures proportional energy savings from around 7%, up to around 25%, w.r.t. BASE.

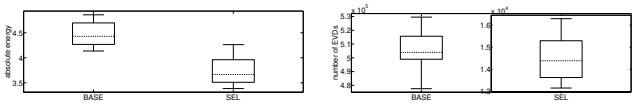
[3]Percentage values are rounded down, *i.e.*, $x\%$ means $\lfloor x\% \rfloor$.

	$c=1$	$c=1.5$	$c=2$	**avg**
$n=50$	6.57%	16.82%	24.39%	**15.93%**
$n=75$	10.61%	19.46%	24.29%	**18.12%**
$n=100$	11.22%	17.37%	18.65%	**15.75%**
avg	**9.47%**	**17.88%**	**22.44%**	**-**

(a) Phase six



(b) Phase seven



(c) Phase eight

Figure 4. Third experiment: comparison between the farthest-node removal policy (BASE) and the selected technique (SEL) in terms of (a) energy costs (expressed as *relative energy gains*) and (b) computational overhead (expressed as relative EVDs). In (c) a comparison in terms of energy and computational overhead is performed over a class of larger networks ($n = 400$ and $c = 1$), by means of boxplots (in the second boxplot, the different y-axes scales should be noticed).

b) Comparing computational overhead (phase seven): in the previous phase, we observed that the selected technique performs favorably w.r.t. the baseline counterpart, in terms of overall energy cost necessary to the optimized network topology (and consequently to the individual nodes of the network) to reach the average consensus. Nevertheless, energy saving of the optimized network topologies is only one of the metrics to consider when tackling such a computationally difficult problem. Indeed, from an optimization perspective, it is also extremely interesting to design techniques able to balance the search capability of the algorithm and the overall computational complexity. In this respect, we cannot forget that the original RA problem is NP-hard [22], so good heuristics able to wisely sample and explore such a complex search space are interesting from a research point of view. To this aim, in Fig. 4(b) we show the relative number of EVDs required by SEL w.r.t. BASE. Specifically, we observe that the former required less than 10% of the total EVDs that were necessary to the latter, to obtain the results of the previous phase. In particular, for SEL we have that $e_w \leq \sqrt{n}$, so $\mathcal{E}_{SEL} \leq M \cdot \sqrt{n}$. Finally, it is also important to notice that, as the network size and density increase (*i.e.*, for increasing values of n and c) the EVD-gain increases. This observation will be the basis of the last phase of this experiment.

c) Comparing techniques on larger network topologies (phase eight): in this final phase, we aim at comparing the performances of SEL against BASE, for a class of larger network scenarios. More in detail, to run this experiment, we set $n = 400$ and $c = 1$ and executed both the techniques, this time using a different stop criterion w.r.t. the one adopted so far. Specifically, instead of executing the direct search for M iterations, we compute the energy cost of the initial topology (*i.e.*, $c_{t=0}$, associated to $\boldsymbol{L}_{t=0}$) and used it as a reference value in the following way: at each iteration t, c_t is compared with c_0 and, if for more than 10 consecutive iterations the former value is larger than the latter one, then the main optimization loop is interrupted. Obviously, different (and more sophisticated) stop criteria could be adopted, while the goal is only to avoid too premature stops without having to fine-tune the M parameter.

By means of boxplot[4] diagrams, Fig. 4(c) depicts a comparison between the baseline and the selected techniques, in terms of absolute energy and number of EVDs. More in detail, when optimizing the topologies belonging to this class of larger networks, we observe that SEL consistently finds better topologies w.r.t. BASE (on average the energy saving of the final topologies is larger than 17%) at a small fraction of the computational cost (on average, SEL requires only the 2% of the EVDs required by BASE).

D. Fourth experiment

The goal of this experiment is to assess the performance of the proposed framework in more realistic settings. To this aim, we adopt the publicly available Intel Lab's WSN deployment as our experimental topology map. This map consists of 2-dimensional relative locations (expressed in meters, at half meter resolution) of 54 sensor nodes, deployed in the Intel Berkeley Research Lab (*i.e.*, an indoor ICT laboratory spreading over a surface area larger than 1300 square meters) in 2004[5]. However, this topology map cannot be considered as an instance of the RGG model. It follows that the energy cost E_c introduced in (12) cannot reliably model this context, mainly because the receiving energy term expressed in (15) assumes randomly and uniformly distributed nodes in the unity square, which is not the case of this topology. Thus, only for this experiment, we opt for using the so called *first order radio model*, appeared for the first time in the WSN context in [36]. According to this model, the energy spent to receive and send a wireless message are respectively:

$$\bar{E}rx = E_{elec} \cdot L \tag{28}$$

$$\bar{E}tx = E_{elec} \cdot L + \epsilon_{amp} \cdot L \cdot r_i^2, \tag{29}$$

where L represents the length in bits of the received/sent message, E_{elec} is a constant term indicating the energy dissipated by the transceiver circuitry to receive or send one bit over the medium and ϵ_{amp} is the energy consumed by the transmit amplifier to reliably send one bit at a distance r_i, when still assuming a quadratic energy loss due to channel transmission. We have verified that, despite its simplicity, this model is still often adopted in the recent literature. Only as an example, the authors of [37] have used the same radio model

[4]For each boxplot, the lowest and the largest values are represented as whiskers, the lower and upper quartiles are shown as a box and the median value is represented by a dashed line.

[5]The interested reader is referred to http://db.csail.mit.edu/labdata/labdata.html.

and actual parameter setup of [36], as we do in this experiment. Specifically we fix L, E_{elec} and ϵ_{amp} to 200 $bits$, 50 nJ/bit and 100 $pJ/bit/m^2$, respectively.

Similarly to Sec. II-C, we express the total energy consumption to reach a consensus $\bar{E}_c$ as the sum of the energy spent by the individual nodes of the network, this time without averaging and by separately taking into account the receiving and transmitting energy terms, $\bar{E}rx_i$ and $\bar{E}tx_i$ respectively. More in detail:

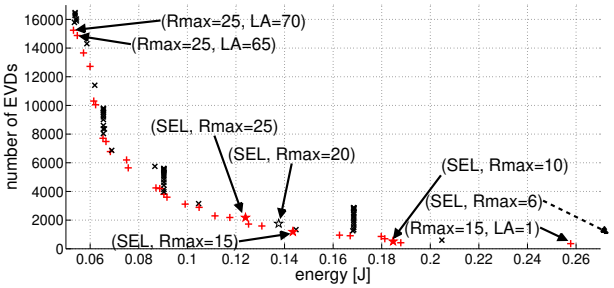
$$\bar{E}_c(\mathcal{R}) = \lceil \tau \rceil \cdot \sum_{i=1}^{n} \left(\overline{\mathcal{N}}_i \cdot \bar{E}rx + \bar{E}tx_i \right). \tag{30}$$

It is worth to notice that, in this case, both the receiving and transmitting energy terms of generic node i are accounted in the total energy cost, while being them dependent on node i's relative position in the topology (and so on $\mathcal{R}$). In fact, $\overline{\mathcal{N}}_i$ represents the number of actual neighboring nodes of node i (as defined in Sec. II-C), while $\bar{E}tx_i$ directly depends on node i's transmission range r_i (as specified in (29)). Notice also that $\bar{E}_c(\mathcal{R})$ is expressed in Joules.

a) Comparing techniques on a class of realistic topologies (phase nine): starting from the Intel Lab's topology map, we have varied the initial common transmission range $r_{\max} \in \{6, 10, 15, 20, 25\}$, hence building a class of 5 realistic topologies[6]. On each topology, we have executed BASE and SEL techniques, as well as a group of quasi-greedy techniques where we fixed the look ahead sizes to 1 (which corresponds to have a purely approximated method) and to $[5:5:100]$[7]. Therefore, the total number of optimization executions constituting our population is 115. For all the members of the population, the value of M was set to infinitive, to ensure optimization executions until topology disconnection; at that time each member returned the best performing topology in terms of $\bar{E}_c$, as described by (30).

To ease the visual analysis of the results obtained by the described population in terms of both energy consumption and computational overhead, Fig. 5(a) depicts a scatter plot with $\bar{E}_c$ and number of EVDs as the x and y axes of the Cartesian plane, respectively. On this plane we projected the results of each member of the population, marking the Pareto-dominant solutions (*i.e.*, the members of the *Pareto front*) with red crosses and the Pareto-dominated ones with black points [38]. Moreover, to easily identify the SEL members of the population, they are marked with stars in the figure (full-red stars for Pareto-dominant solutions, empty-black stars otherwise). It is important to notice that all BASE members were largely dominated by the Pareto front: these solutions were not reported in Fig. 5(a) to avoid using too wide axes ranges, which would have negatively affected figure readability. Observing the graph, we see that (except for the SEL member with $r_{\max} = 20$, which does not belong to the Pareto front) the SEL sub-population belongs to the most interesting portion of the Pareto front (*i.e.*, the area located around the *knee* of the front, where the optimal trade-offs between energy consumption and computational cost are projected).

For the sake of completeness, Table 5(b) summarizes some interesting metrics of the SEL sub-population. In this table, E_i and E_b are energy costs, computed as (30) and expressed in Joules, of the initial and the best topology, respectively; $\rho = E_b/E_i$ (expressed in percentage, where lesser values mean better payoffs of the best topology w.r.t. the initial topology); it_b and it_d represent the optimization iterations at which the corresponding optimization process found the best configuration and the first disconnected configuration, respectively; σ is the *sparsity degree* of the best topology w.r.t. the initial topology, computed as the ratio between their corresponding number of links (expressed in percentage, where higher values mean sparser best topologies); finally, ω represents the relative *overhead degree* of the executor, computed as the ratio between the number of exact the approximated number of EVDs computed by the optimization process to find the best topology (expressed in percentage, where higher values mean higher computational overhead due to exact EVD calculations). These results confirm what highlighted in the previous experiments, that is the sparsification process always lead to better topology configurations (the best topologies are sparser than the original ones, up to 90% of sparsity degree), in terms of energy spent by the nodes of the network to reach the consensus. In detail, the optimized topologies are able to save up to more than the half of the energy required by their initial counterparts. Such benefits are more pronounced when starting from initial dense scenarios. For this reason, it is of key importance to rely on very fast assessment methods. Just as an example, starting from the densest topology (*i.e.*, $r_{\max} = 25$), our approximated method reached the best solution at 96% of the whole optimization process (iteration 826 on a total of 860 iterations), while the whole optimization process required only 0.52% of exact EVDs calculations required to its (hypothetical) exact counterpart.



$r_{\max}$	E_i	E_b	ρ	it_b	it_d	σ	ω
6	0.88	0.68	77.33%	23	37	25.56%	5.89%
10	0.44	0.18	41.64%	140	164	63.93%	2.29%
15	0.29	0.14	49.28%	324	359	78.45%	1.41%
20	0.27	0.14	50.36%	568	600	86.85%	0.82%
25	0.23	0.12	54.81%	826	860	90.57%	0.52%

(b) Phase nine

Figure 5. Fourth experiment: visual comparison of BASE and SEL techniques over a class of realistic topologies generated from the Intel Lab's topology map, in terms of (a) energy (expressed in Joules) versus computational overhead (expressed as number of EVDs). In (b) some interesting metrics of the SEL sub-population are reported.

[6]Notice that, when $r_{\max} = 5$, the initial topology was already disconnected.

[7]A Matlab colon notation is used here to refer to an interval of integer values from 5 to 100, extremes included, with a increment value of 5 units, for a total of 20 integer values.

VI. Conclusions

In this paper, we proposed an *integrated optimization framework* able to reduce a topology-dependent cost function, in the context of broadcast WSNs. This framework, by means of topology design, could effectively improve the convergence speed of an average consensus algorithm, thus reducing the energy consumption of the sensor nodes executing such distributed algorithm. The effectiveness of the proposed framework was thoroughly assessed over various classes of network scenarios. At the end of this wide simulation campaign, we could isolate a specific technique realized with the proposed framework and we compared it against a pure greedy strategy recently proposed in the literature. This comparison revealed consistently better results of the implemented technique over the state-of-the-art technique, both in terms of energy saving at sensor nodes and computational power at central processing unit, evaluated in terms of number of eigenvalue decompositions. With respect to such capability of reducing the required computational power without affecting the quality of the final topology, the selected technique is particularly suitable when the network scenario to optimize increases in size and density.

References

[1] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[2] Q. Shen, P. Shi, Y. Shi, and J. Zhang, "Adaptive output consensus with saturation and dead-zone and its application," *IEEE Transactions on Industrial Electronics*, 2016.

[3] H. Ji, F. L. Lewis, Z. Hou, and D. Mikulski, "Distributed information-weighted Kalman consensus filter for sensor networks," *Automatica*, vol. 77, pp. 18–30, 2017.

[4] S. R. Etesami and T. Başar, "Convergence time for unbiased quantized consensus over static and dynamic networks," *IEEE Transactions on Automatic Control*, vol. 61, no. 2, pp. 443–455, 2016.

[5] C. Asensio-Marco and B. Beferull-Lozano, "Energy efficient consensus over complex networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 2, pp. 292–303, 2015.

[6] A. Khosravi and Y. S. Kavian, "Challenging issues of average consensus algorithms in wireless sensor networks," *IET Wireless Sensor Systems*, vol. 6, no. 3, pp. 60–66, 2016.

[7] B. Dulek, O. Ozdemir, P. K. Varshney, and W. Su, "Distributed maximum likelihood classification of linear modulations over nonidentical flat block-fading gaussian channels," *IEEE Transactions on Wireless Communications*, vol. 14, no. 2, pp. 724–737, 2015.

[8] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[9] J. Qin, W. Fu, H. Gao, and W. X. Zheng, "Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory," *IEEE Transactions on Cybernetics*, vol. 47, no. 3, pp. 772–783, 2017.

[10] Z. Wang, P. Zeng, M. Zhou, D. Li, and J. Wang, "Cluster-based maximum consensus time synchronization for industrial wireless sensor networks," *Sensors*, vol. 17, no. 1, p. 141, 2017.

[11] M. Vazquez-Olguin, Y. Shmaliy, and O. Ibarra-Manzano, "Distributed unbiased fir filtering with average consensus on measurements for wsns," *IEEE Transactions on Industrial Informatics*, 2017.

[12] M. Vecchio, R. Giaffreda, and F. Marcelloni, "Adaptive lossless entropy compressors for tiny IoT devices," *Transactions on Wireless Communications*, vol. 13, no. 2, pp. 1088–1100, 2014.

[13] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[14] 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.

[15] M. E. Chamie, G. Neglia, and K. Avrachenkov, "Distributed weight selection in consensus protocols by schatten norm minimization," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1350–1355, May 2015.

[16] S. Kar and J. M. F. Moura, "Topology for global average consensus," in *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*, 2006, pp. 276–280.

[17] S. Barbarossa, G. Scutari, and A. Swami, "Achieving consensus in self-organizing wireless sensor networks: The impact of network topology on energy consumption," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, vol. 2, 2007, pp. 841–844.

[18] C. Asensio-Marco and B. Beferull-Lozano, "Accelerating consensus gossip algorithms: Sparsifying networks can be good for you," in *IEEE International Conference on Communications (ICC 2010)*, 2010, pp. 1–5.

[19] ——, "A greedy perturbation approach to accelerating consensus algorithms and reducing its power consumption," in *IEEE Statistical Signal Processing Workshop (SSP 2011)*, June 2011, pp. 365–368.

[20] S. Sardellitti, S. Barbarossa, and A. Swami, "Optimal topology control and power allocation for minimum energy consumption in consensus networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 383–399, Jan. 2012.

[21] M. Vecchio and R. López-Valcarce, "A greedy topology design to accelerate consensus in broadcast wireless sensor networks," *Information Processing Letters*, vol. 115, no. 3, pp. 408–413, 2015.

[22] B. Fuchs, "On the hardness of range assignment problems," *Networks*, vol. 52, no. 4, pp. 183–195, 2008.

[23] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "Energy-efficient broadcast and multicast trees in wireless networks," *Mobile networks and applications*, vol. 7, no. 6, pp. 481–492, 2002.

[24] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[25] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on information theory*, vol. 46, no. 2, pp. 388–404, 2000.

[26] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Comput. Surv.*, vol. 37, no. 2, pp. 164–194, 2005.

[27] C. Godsil and G. F. Royle, *Algebraic graph theory*. Springer Science & Business Media, 2013, vol. 207.

[28] R. Hooke and T. A. Jeeves, ""direct search" solution of numerical and statistical problems," *Journal of the ACM*, vol. 8, no. 2, pp. 212–229, 1961.

[29] R. M. Lewis, V. Torczon, and M. W. Trosset, "Direct search methods: then and now," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 191–207, 2000.

[30] D. Bäumer, G. Gryczan, R. Knoll, C. Lilienthal, D. Riehle, and H. Züllighoven, "Framework development for large systems," *Magazine Communications of the ACM*, vol. 40, no. 10, 1997.

[31] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed., ser. Wiley Series in Probability and Statistics. Wiley, 2009.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.

[33] P. Hansen, N. Mladenović, and J. A. M. Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, vol. 175, no. 1, pp. 367–407, 2010.

[34] N. P. van der Aa, H. G. ter Morsche, and R. R. M. Mattheij, "Computation of eigenvalue and eigenvector derivatives for a general complex-valued eigensystem," *Electronic Journal of Linear Algebra*, vol. 16, pp. 300–314, 2007.

[35] J. B. Atkinson, "A greedy look-ahead heuristic for combinatorial optimization: An application to vehicle scheduling with time windows," *Journal of the Operational Research Society*, vol. 45, no. 6, pp. 673–684, 1994.

[36] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. of the 33rd Hawaii International Conference on System Sciences*, vol. 8, 2000, pp. 8020–.

[37] G. S. Arumugam and T. Ponnuchamy, "Ee-leach: development of energy-efficient leach protocol for data gathering in wsn," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, Mar 2015.

[38] M. Voorneveld, "Characterization of Pareto dominance," *Operations Research Letters*, vol. 31, no. 1, pp. 7–11, 2003.

Massimo Vecchio received the M.Sc. degree in Information Engineering (Magna cum Laude) from the University of Pisa and the Ph.D. degree in Computer Science and Engineering (with Doctor Europaeus mention) from IMT Lucca Institute for Advanced Studies in 2005 and 2009, respectively. Starting from May 2015, he is an associate professor at eCampus University, while in September 2017 he has also joined FBK CREATE-NET to coordinate the research activities of the OpenIoT Research Unit. He is the project coordinator of AGILE (www.agile-iot.eu), a project co-founded by the Horizon 2020 programme of the European Union. His current research interests include computational intelligence and soft computing techniques, the Internet of Things paradigm and effective engineering design and solutions for constrained and embedded devices. Regarding his most recent editorial activity, he is a member of the editorial board of the Applied Soft Computing journal and of the newborn IEEE Internet of Things Magazine, besides being the managing editor of the IEEE IoT newsletters.

Gennaro Amendola received the degree and the Ph.D. degree in Mathematics from the University of Pisa (Italy), in 1999 and 2004, respectively. Currently, he is an associate professor at eCampus University (Italy), where he is the president of the Evaluation Board. He is interested in both Pure and Applied Mathematics. His research topics deal with Geometry (more precisely, low dimensional topology) and applications of Mathematics to Social choice, Engineering, Management, Informatics and Geodesy. He has coauthored 13 papers in international journals and conference proceedings.

Pietro Ducange received the M.Sc. degree in Computer Engineering and the Ph.D. degree in Information Engineering from the University of Pisa, Pisa, Italy, in 2005 and 2009, respectively. Currently, he is an associate professor at eCampus University, Novedrate, Italy, where he is the director of the SMART Engineering Solutions & Technologies (SMARTEST) Research Centre. His main research interests include evolutionary fuzzy systems, big data mining, social sensing and sentiment analysis. Further, he has been involved in a number of R&D projects in which data mining and computation intelligence algorithms have been successfully employed. He has coauthored more than 40 papers in international journals and conference proceedings. Prof. Ducange is a Member of the Editorial Boards for Soft Computing and the International Journal of Swarm Intelligence and Evolutionary Computation.