Exploiting synergies between exact and heuristic methods in optimization: an application to the relay placement problem in wireless sensor networks

Eduardo Feo Flushing and Gianni A. Di Caro

Dalle Molle Institute for Artificial Intelligence (IDSIA), Manno-Lugano, Switzerland
{eduardo,gianni}@idsia.ch

Abstract. Exact and heuristic methods for solving difficult optimization problems have been usually considered as two completely different approaches, with each of them being extensively studied by a different research community and used in different fields of application. In this work we propose a scheme for the integration and cooperation of an exact method, based on mixed-integer linear programming (MILP), and a bio-inspired metaheuristic, a genetic algorithm (GA), for the solution of a relay node placement problem in wireless sensor networks.
The integration aspect relates to the use of a MILP solver inside GA's operators. The cooperation aspect is implemented through a shared incumbent environment in which a MILP solver and the GA work in parallel and exchange relevant information in order to overcome their individual weaknesses and provide better solutions within a shorter time.
Experimental results show a significant increase of performance when integration and cooperation take place, in comparison to the performance of the MILP and the GA when used independently from each other.

Keywords: MILP, genetic algorithm, relay placement, hybrid metaheuristic, wireless sensor networks, shared incumbent, optimization

1 Introduction

In this work we implement a novel synergistic search scheme for solving complex optimization problems. Our approach is based on a social metaphor: a community of problem solvers, of possibly different nature and featuring different skills, communicate and cooperate with each other in order to find the best solution to a common problem. The solvers actively and concurrently exploit their mutual differences in order to produce community-level synergies and overcome individual weaknesses. More specifically, we consider the combination of exact and heuristic methods, considering a *mixed-integer linear programming (MILP)* approach and a *genetic algorithm (GA)*, a bio-inspired metaheuristic.

MILP and metaheuristics are two different paradigms for solving optimization problems. The former is characterized by the definition of a mathematical program including linear equations and both continuous and integer variables.

A solution is determined according to methods that intelligently explore the search space and provide a guarantee on the optimality of the solution found. Metaheuristics, on the other hand, aim to exploit problem-specific knowledge and to find good (possibly optimal) solutions within short time, but not providing any guarantees on the quality of the solution found. Given their structural differences, both approaches exhibit complementary properties, such that the strengths of one can help to overcome the weaknesses of the other, and vice versa. For example, the utter exploration of the solution space performed by a MILP solver can be exploited by the metaheuristic search in order to avoid fast convergence to or get trapped in local optima. Also, the MILP solver may use the local optima identified by the metaheuristic search to improve its search bounds and therefore be able to identify areas of the solution space which can be safely excluded from the search. These potential synergies suggest the combination of both approaches to get the best of two worlds. The relatively new domain of research in *matheuristics* [19] precisely addresses the study of the integration of exact and heuristic methods.

The specific optimization problem that we tackle in this paper is the *relay node placement for performance enhancement* problem (RNP-PE in short), a complex combinatorial optimization problem with several applications in Wireless Sensor Networks (WSNs). In general terms, the relay node placement optimization problem can be defined as selecting, from a given set of possible locations, a limited number of positions where some additional nodes (called relay nodes) can be deployed. The objective of the deployment is to improve the network performance metrics of interest. In the case of our RNP-PE, we define these metrics as the *end-to-end communication delays* and the overall data *throughput* obtained at the *sink nodes* of the WSN.

Realistically large instances of the RNP-PE problem present an exponential number of possible deployments. Moreover, for each single deployment, an optimal routing problem must be solved in order to evaluate its quality. Unfortunately, the optimal routing problem belongs to the family of $\mathcal{NP}$-hard network design problems [13,25]. For this reason, standard approaches fail to find optimal solutions to the RNP-PE, even for small size instances, in reasonably short time. To overcome these limitations and find good solutions to the RNP-PE within short time and with quality guarantees, we employ a social approach for problem solving which is based on cooperation and integration. The *cooperation* aspects involve the use of different strategies, namely a MILP solver and a GA heuristic, working in parallel and continuously exchanging relevant search information. The exchanged information is used by each side to improve its performance. The *integration* lies in the decomposition of the problem, and the inclusion of another MILP solver inside the GA. The decomposition approach enables the heuristic to obtain valuable information which is exploited by the genetic operators.

2 Related work

Bio-inspired approaches, and more specifically GAs, have been used to tackle the complexity of many optimization problems in wireless networks, including

clustering [2], optimal design [12], routing and link scheduling [4], network planning [23] and node placement [7, 22, 26, 31, 32]. Despite the success of GAs and other metaheuristics in solving complex optimization problems, there is a growing feeling that pure metaheuristics are reaching their limits [5]. As technology advances, new and increasingly complex optimization problems arise. These observations, together with the exhaustion of new design ideas for pure metaheuristics, is leading the researchers to explore the combination of different techniques and the proposal of hybrid methods [19].

The main motivation behind the hybridization of different algorithms is to exploit their potential synergies. Not until recently, hybrid approaches began to appear in networking optimization problems. In [3], a tabu search is embedded in the genetic operators of a conventional GA to solve a frequency assignment problem, while in [17], tabu search is combined with simulated evolution to tackle a network routing problem. In [26], local search heuristics are used to evaluate GA's individuals to solve a switch location problem in cellular networks.

Within the vast possibilities of hybrid methods for combinatorial and discrete optimization, there is a promising direction of research in combining exact mathematical programming techniques and meta-heuristic approaches. According to the structure of these different solution approaches, two main categories have been identified: *integrative and collaborative combinations*. In integrative combinations, one technique is usually embedded inside another technique, hence the latter is seen as a functional component of the former. On the other hand, collaborative combinations feature two or more methods running sequentially, intertwined or in parallel, which are not part of each other [24].

A *shared incumbent environment* is a general methodology to realize collaborative combinations of mathematical programming and metaheuristic approaches. The main idea of the shared environment is to allow both components (i.e., a MILP solver and a metaheuristic algorithm) to exchange information about their current best known solutions. This information is used by the MILP solver to improve its current incumbent and prune the branch and bound tree, and by the meta-heuristic to guide the search to more promising regions of the solution space and to prevent getting stuck in local minima [28].

Compared to previous approaches, our work exploits the synergies between different methods in two dimensions both in integration and cooperation. We consider a heterogeneous group of solvers, composed of standard mathematical solvers and a well-known bio-inspired metaheuristic.

3 The problem

A *wireless sensor network* (WSN) consists of a set nodes that are equipped with sensing and limited processing capabilities, and that can locally communicate with each other through a wireless medium [1]. The *sensor nodes* (SNs) composing a WSN are usually inexpensive and low-powered, such as they can be deployed in large numbers to provide monitoring and sensing services for long time periods. In typical applications, the data generated by the sensor nodes

need to be transmitted to and aggregated and processed at *base stations* (BSs). The general model for the forwarding of the data from SNs to BSs is based on the definition and the use of *multi-hop routing paths*.

Since a WSN can operate for relatively long times and/or it can be embedded in dynamic or hostile environments, a core issue in WSNs is the definition of effective strategies to maintain the network aoperational for long time period and/or for its adaptation to external or internal changes. In this direction, a wealth of research has considered the use of special nodes, referred to as *relay nodes* (RNs), that can be deployed and added to the WSN after the network has been put in place. RNs can be positioned by hand, or can be part of a mobile robotic unit, such that they can be deployed autonomously or on-demand. Possible roles of RNs include the provisioning of connectivity [6,9,18,20,27], *extend the network lifetime* [15,30], *energy-efficient or balanced data gathering* [10,21], and to provide *survivability and fault tolerance* [14,16,20,27,33].

The *relay node placement for performance enhancement* problem is defined as follows: given a set of possible locations where to deploy a restricted number of available RNs, we aim to select from this set the locations in which the additional nodes can be positioned to improve throughput and end-to-end packet delays for the data gathered at BSs. Although the primary objective is determining the physical locations where RNs should be placed to, the solution also specifies the way these RNs should be used. This specification comes in the form of *optimal routing paths* from SNs to BSs to forward the data flows. We assume that the initial WSN is connected, therefore the use of RNs is entirely devoted to improve the performance of the network. We present the RNP-PE as a *linear, mixed integer mathematical program* (MILP). The formulation includes a number of constraints and penalty components, aimed at closely modeling the specific characteristics of the wireless environment.

3.1 MILP Model

We model the WSN as a set of SNs and BSs located in a set of known positions $\mathcal{S}$ and $\mathcal{B}$, respectively. SNs both generate and forward data packets towards one of the BSs in multi-hop fashion (a data flow can be split over multiple paths). We assume that the characteristics of data generation characteristics for each SN are known. All nodes communicate with each other within the *communication range* $\mathbf{r}$. A set of K RNs is also available, their role is to forward data received from other nodes. The placement of node relays is restricted to a *numerable set of candidate locations* denoted as $\mathcal{R}$. We formalize the RNP-PE by a MILP model based on a *minimum cost flow* formulation as follows.

Let $G = (V, E)$ be a connected digraph representing a WSN, where $V = \mathcal{N}$ is the set of nodes, and E is the set of communication links. $\gamma : E \mapsto \Re$ is a *link cost* function, and $\tau : \mathcal{S} \mapsto \Re$ is a data generation (*traffic load*) function, expressed in the *data per second* generated by an SN. In the following, we measure τ in terms of *flow units*, f_{unit}, expressed as bytes/sec. Data flows and relay positions define the two sets of *decision variables*. The *flow variable* f_{ij} denotes the amount of flow through link (i, j), that is, the data traffic to be sent from node n_i to node n_j

located at positions $i,\ j \in \mathcal{N}$ respectively. f_{ij} values are expressed in *flow units*. The *binary positional variable* y_i indicates whether location $i \in \mathcal{R}$ is used to circulate flow or not. When y_i is set to 1 in a solution, an RN is to be positioned at the corresponding relay location. A full solution specifies both flows and relay positions. The SN-to-BS routes are defined in the *routing-tree* induced by the set $\{(i,\ j) \in E \mid f_{ij} > 0\}$. The complete MILP model is presented in Figure (1).

$$\text{min RNP-PE} = \sum_{(i,\ j)\in E} \gamma_{ij} f_{ij} + \hat{R} \sum_{i\in\mathcal{R}} y_i + \alpha \sum_{i\in\mathcal{S}} p_i \hat{F}\ . \tag{1}$$

$$\text{subject to:} \quad \sum_{(i,\ j)\in E} f_{ij} - \sum_{(j,\ i)\in E} f_{ji} = \begin{cases} \tau_i & \text{if } i \in \mathcal{S}, \\ 0 & \text{if } i \in \mathcal{R} \end{cases} \tag{2}$$

$$\sum_{i\in\mathcal{B}} \sum_{(j,\ i)\in E} f_{ji} = \sum_{k\in S} \tau_k \tag{3}$$

$$y_i = 1 \iff \sum_{j\in\mathcal{N}} f_{ji} > 0 \quad \forall i \in \mathcal{R} \tag{4}$$

$$\sum_{i\in\mathcal{R}} y_i \leq K \tag{5}$$

$$\sum_{(i,\ j)\in E} f_{ij} + \sum_{(j,\ i)\in E} f_{ji} \ \leq\ L_{cap}, \quad \forall i \in \mathcal{N} \tag{6}$$

$$b_{ij} = 1 \iff f_{ij} > 0 \quad \forall i,\ j \in \mathcal{N} \tag{7}$$

$$\sum_{(j,\ i)\in E} b_{ji} \leq D \quad \forall i \in \mathcal{S} \tag{8}$$

$$p_i = 1 \iff \sum_{(i,\ j)\in E} \sum_{(j,\ k)\in E} f_{jk} \geq \bar{F}_{max} \tag{9}$$

Fig. 1: MILP formulation of the RNP-PE problem.

Constraints (2-3) correspond to the flow definition. The number of available RNs is limited to K constraints (4-5). Given that the optimal solution can correspond to a number of RNs $k < K$, we define a penalty factor in the objective (1) to favor the use of a minimal amount of RNs: any optimal solution using n relays needs to provide a minimal gain $\hat{R}$ with respect to the solution obtained using $n-1$ relays. Parameter $\hat{R}$ can be adjusted according to the problem instance (e.g., relay node availability, economic cost).

Shared wireless channels in WSNs are necessarily *bandwidth-limited*. This condition is reflected by *link capacity* parameter L_{cap}, which is the nominal amount of data (bytes/sec) that can be transmitted by a wireless link in the network (assuming the same capacity for all links), and constraint (6).

For a node n, the *routing in-degree* is the number of n's neighbors using n to relay data. Because of shared medium and contention access, this number strongly impacts on the effective node capacity and network load balancing. Hence, node degree in the routing trees are limited by constraints (7 - 8).

To minimize wireless interference and produce balanced routing trees, which allow balanced energy depletion, we need to setup *minimally interfering flow*

paths. We enforce this by including in the objective function a penalty component based on the *maximum local flow*, $\bar{F}_{max}$, defined as the maximum amount of flow that can circulate within a disk of radius $\mathbf{r}$ centered on an SN. The calculation of the flow circulating within the $\mathbf{r}$-disk of an SN i, requires to sum up the outgoing flows from all i's neighbors. This notion is included in constraints (9), where p_i is a binary penalty variable that takes value 1 when the flow through the i's $\mathbf{r}$-disk violates the maximum allowed amount. In order to use p for inclusion in the objective function as penalty, we derive a rough estimation, $\hat{F}$, of the optimal solution value of problem, without penalties, and we use it as a penalty score for the violation of the circulating flow limit. Using $\hat{F}$ and p, the penalty for the violation in maximum local flow is therefore included in the objective function. The parameter α weighs the penalty in the objective function. In the experiments we set $\alpha = 0.1$. We refer the interested reader to [8, 11] for a full description of the parameters and an extensive evaluation of the RNP-PE model.

4 Integration: A hybrid genetic algorithm

In the RNP-PE formulation, it is primarily the number of possible assignments of relay positions which affects the size of the problem. In instances where $|\mathcal{R}|$ is in terms of thousands (which is a typical scenario), even for small number of relays to select, the solution space is exponentially large.

Considering that the RNP-PE jointly solves two problems, namely the node placement and data routing, we decompose the problem in two hierarchical levels. At the top level, the GA is used to explore possible relay placements, which involves iterating over assignments of the variables y_i of the full MILP model. At the bottom level, a *simplified* MILP is used to compute the optimal routing of each one of these placements. This means that, for a certain assignment of y_i (i.e., positioning of relays), a MILP solver computes the best possible flow routing scheme achievable (based on the RNP-PE formulation). The optimization problem obtained by fixing the variables y_i is much smaller and simpler than the original one, therefore its computation time is much reduced, a property which is effectively exploited by the decomposition method.

4.1 Solution encoding

The encoding of individuals (also known as *chromosome encoding*) is fundamental to the implementation of GAs in order to efficiently transmit the genetic information from parents to offsprings. In our case, an individual of the population represents a deployment of relay nodes. Since RNs can be placed at one of the set of candidate locations $\mathcal{R}$, the location of each RN can be conveniently specified as the index of the element in $\mathcal{R}$ to which it corresponds to. Accordingly, a population member is encoded as an array of indexes values. The array size is variable, given that the problem can admit solutions with $k \leq K$ relays.

Apart from the encoding, another fundamental aspect of a GA is the design of its genetic operators. We propose a set of application-specific genetic operators designed to achieve a good trade-off between exploration and exploitation.

4.2 Evaluation of individuals

Let $R^* \subseteq \mathcal{R}$, be an individual of the genetic population, with $|R^*| \leq \mathcal{K}$. In order to evaluate the quality of R^*, we compute the optimal routing scheme based on the relays located at the positions in R^*. A simple way to perform this computation consists in constructing and solving a MILP problem, using the RNP-PE formulation and replacing $\mathcal{R}$ with R^*. However, in practice, this approach involves an additional overhead in terms of computation time, since a complete mathematical model (which needs to be constructing and initialized) must be built each time. The overhead is more evident after the evaluation of hundreds (or even thousands) of individuals, as usually performed by a GA.

To overcome this issue, we propose to initially build a complete model, using the whole set $\mathcal{R}$, and add and remove constraints to the original formulation every time an individual need to be be evaluated. This way turns out to be both efficient and time-saving. To evaluate R^* using the original formulation presented in section (3.1), the following constraints must be included:

$$y_i = 1 \quad \forall i \in R^* \tag{10}$$

$$y_i = 0 \quad \forall i \in \mathcal{R} \setminus R^*. \tag{11}$$

Additionally, since the best data routing schemes may be achieved using less than $|R^*|$ relays, the model need to be enabled to obtain solutions in which not all of the positions in R^* are used for data *flows*. However, the original formulation enforces the generation of routing solutions such that, if y_i is equal to 1, some flow variable in the set $\{f_{ji},\ j \in \mathcal{N}\}$, must be positive. This implies that by adding constraints (10), we are forcing the model to produce routing path that require to use all the positions specified in R^*. To avoid this undesirable behavior, we relax the constraints (4) by replacing them with:

$$y_i = 1 \Leftarrow \sum_{j \in \mathcal{N}} f_{ji} > 0 \quad \forall i \in \mathcal{R}. \tag{12}$$

After solving the now simplified MILP, the obtained solution is checked for *unused* relays, that is, for positions $i \in R^*$ such that $\sum_{j \in \mathcal{N}} f_{ji} = 0$. For these positions, the objective function value is recomputed assuming $y_i = 0$, hence obtaining an accurate assessment of the optimal routing paths.

4.3 Routing-aware placement (RAP) crossover

We propose an application-specific crossover operator, referred to as *Routing-aware placement* (RAP) crossover, aimed to boost the performance of the GA search. The main ideas behind our operator are the design of a multi-parent crossover operation and the inclusion of the extracted information from the routing trees obtained by the evaluation of previous individuals.

During the progress of the GA, the routing trees obtained after the evaluation of each solution are examined and analyzed to extract useful information about the shape of the generated sub-optimal solutions. In these routing trees,

some relay nodes (and hence their corresponding positions) are more important than others, in the sense that they are used to carry more data. Therefore, we measure the *degree of importance* of relay positions by looking at the amount of flow circulating through the positioned relay nodes each time we evaluate an individual. The most important relay positions form a set which we refer to as ***preferential relay set***, hereafter denoted as $PS \subseteq \mathcal{R}$.

Since close-by relay positions are likely to be used in a similar manner, the set PS might become a group of *clustered relay positions*. To promote diversity in PS, we propose a strategy which consists on partitioning the area into a number of regions, where to each region is assigned a quota in PS, corresponding to the maximum number of relay positions located in that region. The maximum size of PS is limited by a user-defined parameter (set to $2K$ in this paper).

Another structural property that we observe in the routing trees, is the presence of ***relay chains***. That is, the combined use of relays forming chains to link different regions of the network. A pair of *chained relay positions* appears in a solution whenever RNs located at these positions are linked to each other, meaning that data flows directly from one relay to the other. After every solution evaluation, we extract all pairs of chained relay positions and store them inside a map structure, $CR : \mathcal{R} \mapsto 2^{\mathcal{R}}$, where the set $CR(i) \subseteq \mathcal{R}$ contains all the positions that have formed a chain with i.

Finally, we also identify possible ***conflicts*** between relay positions. A conflict is identified whenever a relay position is included as an individual but is not used to carry data in the routing solution. This situation suggests that the potential benefits of using the ignored relay position have been achieved by other RNs, placed in different positions. Hence, we say that the ignored relay positions are *in conflict* with the currently utilized ones, and vice versa. To this end, we keep a conflict map $CM : \mathcal{R} \mapsto 2^{\mathcal{R}}$ that associates to each RN position a set of positions that appear to be in conflict with any of the routing tree solutions. Figure (2) illustrates the analysis of the routing solutions performed after the evaluation of an individual. In the example, one pair of chained relays, one preferential relay position (carrying data from four static nodes), and one conflict (unused position) are detected in the routing tree. The description of the algorithmic procedure of the RAP crossover is presented in Figure 1. The parameter $pChainedRelay$ regulates the use of chained relay positions. We denoted as Mom and Dad the genetic parents. The size of the new child individual is taken randomly in the interval defined by the size of both parents involved in the operation. The operator first tries to construct a conflict-free set of RN positions by randomly taking genes from both parents. Each time a gene g is selected, it also takes a chained relay from $CR(g)$ with probability $pChainedRelay$. After this attempt to construct a child, the number of selected genes might not be enough to achieve the target child size. Therefore, a second step is performed in which the remaining RN positions are randomly selected from the preferential set, and in case there are not enough preferential relays, the remaining genes are selected taken randomly from $\mathcal{R}$. In this way, the operator produces a new chromosome which shares genetic material from both parents and from the set

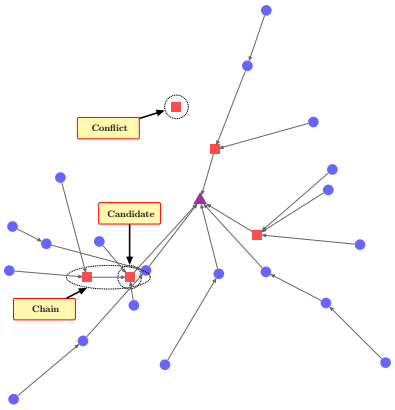


Fig. 2: Elements extracted from MILP solution

PS. Moreover, the RAP crossover operator reinforces the generation of better individuals by excluding conflicts previously seen, replacing those conflicts by potentially useful relays (i.e., from PS), and by introducing chained relays positions, which are usually very beneficial, but not likely to appear frequently when using a completely random procedure.

4.4 Mutation

The proposed mutation operator allows a controlled exploration of new regions of the solution space by inducing small perturbations to existing individuals. Mutation is implemented by displacing the relay positions within a circular area of size 2**r**. Apart from changing the positions of the relays, the mutation may also modify the size of the solution, that is, the number of positions of relays. With a given probability $pSizeChange$, the operator varies the size, removing existing positions or adding new ones.

5 Cooperation: A shared incumbent environment

At this point, we are able to obtain solutions to the RNP-PE in two different ways: (a) using a standard mathematical solver to solve the MILP model presented in Section (3.1), and/or (b) using the metaheuristic procedure introduced in Section (4). Both approaches offer different advantages and disadvantages. Approach (a) offers the possibility (if we are lucky enough) to find an optimal solution. However, depending on the problem instance, we might run out of computational resources before obtaining a good (or even at least any) feasible solution. Approach (b) offers solutions whose quality depends on the amount of time we provide to the GA procedure. In some cases, the evolution process can get stuck and no improvements can be made to the current solution. Moreover, even though we can get optimality bounds for the provided solutions, these

Input: Parents: $Mom, Dad \subseteq \mathcal{R}, |Dad| \leq |Mom| \leq K$
Input: Preferential relay set: $PS \subseteq \mathcal{R}, |PS| \leq 2K$
Input: Chained relays: $CR : \mathcal{R} \mapsto 2^{\mathcal{R}}$, Conflict map: $CM : \mathcal{R} \mapsto 2^{\mathcal{R}}$
Result: $Child \subseteq \mathcal{R}$
if $Mom = Dad$ **then**
  $Child \longleftarrow RandomChild()$ **return** Child
end
$childSize = RandInteger\,(|Dad|, |Mom|)$
$genePool = Mom \cup Dad$
$Child = \emptyset$
while $|Child| < childSize$ **and** $genePool \neq \emptyset$ **do**
  pick random relay $g \in genePool$ **if** $\exists \hat{g} \in Child : \; g \in CM(\hat{g})$ **then**
    $genePool \longleftarrow genePool \setminus \{g\}$
  else
    $Child \longleftarrow Child \cup \{g\}$ $genePool \longleftarrow genePool \setminus \{g\}$ **if** $Rand(0,1) \leq pChainedRelay$ **and** $|Child| < childSize$ **then**
      pick random relay $h \in CR(g)$ $Child \longleftarrow Child \cup \{h\}$ $genePool \longleftarrow genePool \setminus \{h\}$
    end
  end
end
if $(childSize - |Child|) \leq |PS|$ **then**
  while $|Child| < childSize$ **do**
    pick random relay $g \in PS$ $Child \longleftarrow Child \cup \{g\}$
  end
else
  $Child \longleftarrow Child \cup \; PS$ **while** $|Child| < childSize$ **do**
    pick random relay $g \in \mathcal{R}$ $Child \longleftarrow Child \cup \{g\}$
  end
end
return $Child$

Algorithm 1: RAP crossover operator

bounds might not be tight. Hence, we won't be able to have a strong assessment about the quality of the obtained solution (even if the solution is in fact optimal).

A *cooperative environment* is implemented by the execution of both approaches as two independent processes able to *communicate* with each other. We use a shared incumbent environment as the cooperation scheme between both solvers, which consists in letting both methods continuously exchange their best found solutions so far. In the MILP solver, this corresponds to the best upper bound (also known as the *incumbent solution*). In the GA, it is simply represented by the best individual that has been evaluated so far.

The hybrid nature of the GA and its MILP functional component used in the evaluation operator, facilitates the implementation of the shared incumbent environment. Each time an individual is evaluated by the GA, we can easily obtain, from the reduced MILP, the values of the variables corresponding to the optimal routing solution. These values, together with those corresponding to the positions (which are available from the encoding), allows to compose the solution vector of the complete MILP model. Due to the potentially large number of variables, we use a data compression scheme for the communication of the solution vectors. A graphical representation of the proposed bio-inspired shared environment is presented in Figure (3).

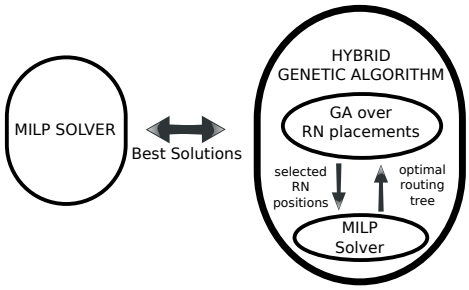
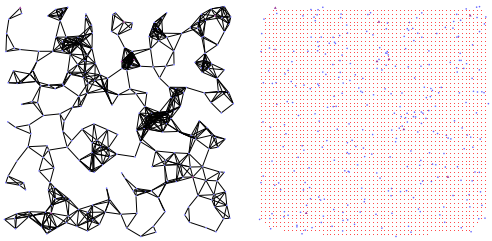


Fig. 3: Cooperative environment based on the shared incumbent.

6 Evaluation

To evaluate the proposed method, we considered a number of randomly generated network instances representing complex scenarios of the RNP-PE. In total, we considered 20 network instances generated with different topological characteristics, i.e. *uniform, clustered*, and *small world*. Networks were embedded in an area of size 150m×150m, and the set of possible relay positions was determined using a uniform grid, with the grid points separated by Δ=2 m of distance. Considering the grid resolution and the size of the area, the number of possible relay positions becomes particularly large (up to 5000). We have considered two group of experiments varying the value of $\mathcal{K}$ since this parameter determines the number of deployments, and has an impact on the performance of the GA. Table (1a) indicates the parameters of network topologies considered. Figure (4) depicts one of the topology instances. The evaluation consisted on three steps.



(a) Initial network (b) Relay locations

Fig. 4: Example of network topology instance.

First, all instances are exhaustively solved using the MILP model presented in Section (3.1). To solve the MILP we used the CPLEX® solver under its default parameters. In this step, the mathematical solver was given a larger amount of computational resources, in terms of both CPU time (10 hours) and physical memory (8 GB RAM), in comparison to the following experiments. In this way,

we aimed at obtaining the optimal or best sub-optimal solutions to each of the problem instances, through an intensive and exhaustive solving process.

In the second step, we solved the same instances using the hybrid Genetic Algorithm presented in section (4). To evaluate the efficacy of the genetic operators, we also use a simplified version of the GA, in which the crossover and mutation operators were replaced by a one-point random crossover and a uniform random mutation operators. In the implementation, we use a steady state genetic algorithm, and the *GAlib library* [29]. The main parameters of the GA are listed in Table (1b). Finally, in the last step, we solve the instances using the cooperative environment as described in Section (5). Both components (i.e., the mathematical solver and the GA) were executed on similar CPUs, and the communication was implemented using sockets. All methods, except the initial step, were given a maximum CPU time of 2 hours and 2GB of physical memory.

Table 1: Experimental setup

(a) Instances parameters

Area	150×150 m^2
$\lvert\mathcal{S}\rvert$	200
$\lvert\mathcal{B}\rvert$	5
TX range ($\mathbf{r}$)	10m

(b) GA parameters

Population size	100
Scaling scheme	Linear
Selection scheme	Tournament
Crossover probability	0.9
Mutation probability	0.1

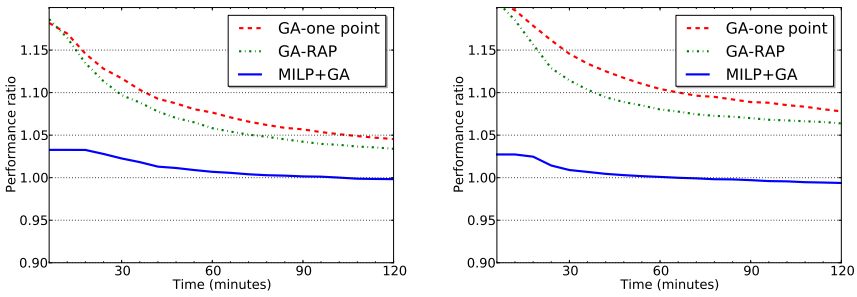
6.1 Experimental results

After performing the first step of the evaluation, we obtained solutions to each of the instances considered. Table (2) shows objective function values, optimality gap, and solving time for some of the instances considered. The mathematical solver was not able to prove the optimality of any of the considered instances. For some instances, the solver ran out of memory and finished before consuming the available CPU time, as noted in the table. In the second step, we analyze performance of the solution approaches, namely the GA with one-point crossover and uniform mutation (GA-one point), the GA with the RAP crossover and the proposed mutation operators (GA-RAP), and the cooperative environment featuring a MILP solver and the GA within a shared incumbent environment (MILP+GA). To measure the performance of each approach, we consider the ratio between the objective function value of their best solutions found and the solutions obtained in the first step. We refer to this value as the *performance ratio*. For each instance, we performed 20 independent runs (to account randomness in the GA procedure), and we took the median value of the performance ratio at different points of time.

Figure (5) shows the average performance ratio over all the instances considered for each value of $\mathcal{K}$. We can observe a variation of performance depending on the choice of the genetic operators. The GA-RAP method is more effective than its naive counterpart and provides better solutions. However, both tend to converge fast and have difficulties in improving their best solution as time

Table 2: Solutions obtained by the MILP.

Obj. value	Gap (%)	CPU time (s)
1576.79	11.79	30290.7
2111.38	13.87	36002
1403.98	16.52	19397.2
1318.45	9.53	16725.7



(a) $\mathcal{K} = 10$ (b) $\mathcal{K} = 20$

Fig. 5: Comparison of the proposed solution methods

advances. On the other hand, the performance of the MILP+GA cooperative scheme is considerably better, and in some of the instances is also able to provide solutions better than those obtained with the first step.

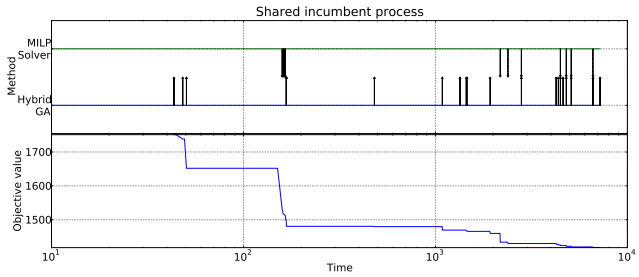


Fig. 6: Illustration of cooperation process.

To illustrate the interactions between both methods within the cooperative environment, we selected one execution of the MILP+GA method. Figure (6) shows the interactions and the evolution of the best solution over time (logarithmic scale). We can observe two clear behaviors: the reactive interactions (sequences of solution exchanges separated by short intervals, and stand-alone interactions. We conclude that none of the methods strictly dominates the others, and the performance obtained by the combination is much superior than the performance achieved by each method individually.

7 Conclusions

We considered an integrative and cooperative environment for solving a complex mixed-integer optimization problem in WSNs. The approach features the use of two problems solvers, of different nature and with orthogonal properties, which interact and cooperate in order to improve their joint performance. Specifically, we presented the case of a bio-inspired metaheuristic (i.e., a genetic algorithm) and a traditional MILP solver for the relay node placement problem in WSNs.

The contributions of this work lie along three lines/ First, we propose a GA which makes use of a decomposition strategy and a MILP solver to evaluate the individuals of the population. Secondly, we designed a cooperative environment in which a MILP solver and the proposed GA work, solve the same problem in parallel, interacting and continuously exchanging relevant information. The objective is to help ach other to overcome their weaknesses and speed-up the solving process. Finally, we demonstrate through extensive experiments the effectiveness of the proposed strategy and show a significant performance improvement when the solvers are combined, compared to their use as independent solvers.

Future work involves considering larger groups (three or more) of solvers, including other types of metaheuristics and mathematical solvers, and studying and assessing the collaboration level achieved according to the size and nature of the group of solvers. Together with this, we will also consider the application of this strategy to other, possibily related, optimization problems.

References

1. Akyildiz, I.: Wireless sensor networks: A survey. Computer Networks 38(4), 393–422 (2002)
2. Al-Obaidy, M., Ayesh, A., Sheta, A.F.: Optimizing the communication distance of an ad hoc wireless sensor networks by genetic algorithms. Artificial Intelligence Review 29(3-4), 183–194 (2009)
3. Alabau, M., Idoumghar, L., Schott, R.: New hybrid genetic algorithms for the frequency assignment problem. IEEE Trans. on Broadcasting (2002)
4. Badia, L., Botta, A., Lenzini, L.: A genetic approach to joint routing and link scheduling for wireless mesh networks. Ad Hoc Networks 7(4), 654–664 (2009)
5. Blum, C., Puchinger, J., Raidl, G.R., Roli, A.: Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing 11(6), 4135–4151 (2011)
6. Cheng, X., Du, D.Z., Wang, L., Xu, B.: Relay sensor placement in wireless sensor networks. Wireless Networks 14(3), 347–355 (2007)
7. Chu, S., Wei, P., Zhong, X.: Deployment of a Connected Reinforced Backbone Network with a Limited Number of Backbone Nodes. IEEE Trans. on Mob. Comp. pp. 1–14 (2012)
8. Di Caro, G.A., Feo, E.: Optimal relay node placement for throughput enhancement in wireless sensor networks. In: Proc. of the 50th FITCE Int. Congress (2011)
9. Efrat, A., Fekete, S., Gaddehosur, P., Mitchell, J., Polishchuk, V.: Improved approximation algorithms for relay placement. Algorithms–ESA (2008)
10. Ergen, S., Varaiya, P.: Optimal placement of relay nodes for energy efficiency in sensor networks. In: Proc. of IEEE ICC. vol. 8, pp. 3473–3479 (2006)
11. Feo, E.: Optimal relay node placement for throughput enhancement in wireless sensor networks. Master's thesis, Trento, Italy (December 2010)

12. Ferentinos, K., Tsiligiridis, T.: Adaptive design optimization of wireless sensor networks using genetic algorithms. Computer Networks 51(4), 1031–1051 (2007)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1990)
14. Han, X., Cao, X., Lloyd, E., Shen, C.: Fault-tolerant relay node placement in heterogeneous wireless sensor networks. IEEE Trans. Mobile Comput. 1 (2009)
15. Hou, Y., Shi, Y., Sherali, H., Midkiff, S.: Prolonging sensor network lifetime with energy provisioning and relay node placement. In: Proc. of IEEE SECON'05. pp. 295 – 304 (2005)
16. Kashyap, A., Khuller, S., Shayman, M.: Relay placement for higher order connectivity in wireless sensor networks. In: Proc. of IEEE INFOCOM. pp. 1–12 (2006)
17. Khan, S.A., Baig, Z.A.: A Simulated Evolution-Tabu search hybrid metaheuristic for routing in computer networks. In: Proc. of IEEE CEC. pp. 3818–3823 (2007)
18. Lloyd, E., Xue, G.: Relay node placement in wireless sensor networks. IEEE Trans. Comput. 56(1), 134–138 (2007)
19. Maniezzo, V., Stützle, T., Voß, S. (eds.): Hybridizing Metaheuristics and Mathematical Programming, Annals of Information Systems, vol. 10. Springer (2009)
20. Misra, S., Hong, S.D., Xue, G., Tang, J.: Constrained relay node placement in wireless sensor networks: Formulation and approximations. IEEE/ACM Transactions on Networking 18(2), 434–447 (2010)
21. Patel, M., Chandrasekaran, R., Venkatesan, S.: Energy efficient sensor, relay and base station placements for coverage, connectivity and routing. In: Proc. of IEEE IPCCC. pp. 581–586 (2005)
22. Perez, A., Labrador, M., Wightman, P.: A multiobjective approach to the relay placement problem in WSNs. In: Proc. of IEEE WCNC. pp. 475–480 (2011)
23. Pries, R., Staehle, B., Staehle, D., Wendel, V.: Genetic algorithms for wireless mesh network planning. In: Proc. of the 13th ACM MSWIM. p. 226 (2010)
24. Raidl, G., Puchinger, J.: Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In: Hybrid Metaheuristics, Studies in Computational Intelligence, vol. 114, pp. 31–62. Springer (2008)
25. Ravi, R., et al.: Approximation algorithms for degree-constrained minimum-cost network-design problems. Algorithmica 31(1), 58–78 (2001)
26. Salcedo-Sanz, et al.: Optimal switch location in mobile communication networks using hybrid genetic algorithms. Applied Soft Computing 8(4), 1486–1497 (2008)
27. Tang, J., Hao, B., Sen, a.: Relay node placement in large scale wireless sensor networks. Computer Communications 29(4), 490–501 (2006)
28. Toklu, N.E., Montemanni, R., Di Caro, G.A., Gambardella, L.M.: A shared incumbent environment for the minimum power broadcasting problem in wireless networks. In: Proc. of the ICICN 2012. IACSIT Press, Singapore (2012)
29. Wall, M.: GAlib: A C++ library of genetic algorithm components. Mechanical Engineering Department, Massachusetts Institute of Technology (1996)
30. Wang, G., Huang, L., Xu, H., Li, J.: Relay node placement for maximizing network lifetime in wireless sensor networks. In: Proc. of IEEE WiCOM. pp. 1–5 (2008)
31. Xhafa, F., Sánchez, C., Barolli, L.: Genetic Algorithms for Efficient Placement of Router Nodes in Wireless Mesh Networks. In: Proc. of the IEEE AINA. pp. 465–472 (2010)
32. Youssef, W., Younis, M.: Optimized asset planning for minimizing latency in wireless sensor networks. Wireless Networks 16(1), 65–78 (2008)
33. Zhang, W., Xue, G., Misra, S.: Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms. In: Proc. of IEEE INFOCOM'07. pp. 1649–1657 (2007)