Applied Soft Computing 111 (2021) 107659

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Solving the regenerator location problem with an iterated greedy approach

Juan David Quintana, Raul Martin-Santamaria, Jesus Sanchez-Oro *, Abraham Duarte

Dept. Computer Sciences, Universidad Rey Juan Carlos, C/. Tulipán, s/n, Móstoles, 28933 (Madrid), Spain

ARTICLE INFO

Article history:
Received 12 November 2020
Received in revised form 21 June 2021
Accepted 24 June 2021
Available online 29 June 2021

Keywords:
Regenerator location problem
GRASP
Iterated greedy
Metaheuristic

ABSTRACT

The evolution of digital communications has resulted in new services that require from secure and robust connections. Nowadays, a signal must be transmitted to distant nodes, and the quality of the signal deteriorates as the distance between the endpoints increases. Regenerators are special components that are able to restore the signal, in order to increase the distance that the signal can travel without losing quality. These special components are very expensive to deploy and maintain and, for this reason, it is desirable to deploy the minimum number of regenerators in a network. We propose a metaheuristic algorithm based on the Iterated Greedy methodology to tackle the Regenerator Location Problem, whose objective is to minimize the number of regenerators required in a network. The extensive computational experiments show the performance of the proposed method compared with the best previous algorithm found in the state of the art.

1. Introduction

Digital communications are without a doubt an ever increasing aspect of our day to day lives. Due to their fundamental place, we must strive to improve their reliability, which can prove challenging specially if we take into account that the communication channel must be able to transmit signals through long distances. Since the signal quality degrades as the distance between the communicating devices increases, a possible solution to this problem is given by the use of regeneration technology.

The regeneration of an optical signal is performed by using a special component, called regenerator, which is able to restore the signal to its initial quality [1,2]. The main drawback of these components is their price, as they are quite expensive. The Regenerator Location Problem (RLP) [3] addressed in this work is intended to minimize the number of regenerators needed to maintain the quality of the signal transmission in a network between each pair of nodes.

A signal in a network can be transmitted between two nodes without quality loss if and only if the length of the shortest path between them does not exceed a maximum distance $d_{\max}$. Otherwise, it is necessary to add one or more regenerators in the path to guarantee signal transmission. Therefore, the shortest path between two nodes is said to be feasible if it does not contain any subsequence of edges with length larger than $d_{\max}$ without internal regenerators. The RLP then consists in determining the minimum number of regenerators that must be deployed in the network such that every pair of nodes is connected through a feasible path.

Fig. 1 presents an example of a network with six nodes and eight links, where the number over each link represents the distance between its endpoints. Considering $d_{\max} = 200$ in this example, it is easy to see a signal can be transmitted from A to C (through B), since the distance is equal to 150, which is smaller than $d_{\max}$. However, it is not possible to transmit a signal from C to E or F, since the length of the shortest path is 225 in both cases. In the last example path of Fig. 1, the shortest path from B to E is B-D-E, with a length of 125, smaller than $d_{\max}$. Feasible paths are represented with a (✓) while those whose length exceeds $d_{\max}$ are represented with a (✗). It is worth mentioning that in this example we only represent 4 shortest paths, while in a network with 6 nodes we can find up to 15 shortest paths to connect each pair of nodes.

In the network depicted in Fig. 1, there are only two pairs of nodes that are not able to communicate without exceeding $d_{\max}$. Specifically, the shortest distance between C and E is 225, resulting from edges C-B (100), B-D (50), and D-E (75). On the other hand, the distance between C and F is also 225, derived from the edges C-B (100) and B-F (125). In order to transmit a signal without losing quality between these nodes it is necessary to insert one or more regenerators in the corresponding path.

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: juandavid.quintana@urjc.es (J.D. Quintana), raul.martin@urjc.es (R. Martin-Santamaria), jesus.sanchezoro@urjc.es (J. Sanchez-Oro), abraham.duarte@urjc.es (A. Duarte).

https://doi.org/10.1016/j.asoc.2021.107659

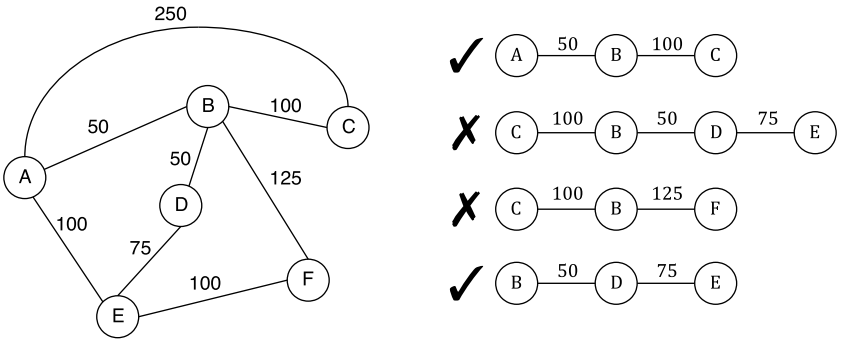

Fig. 1. Example of a network where $d_{\max} = 200$. Some example paths are depicted on the right, with a (✓) if it is feasible or (✗) otherwise.

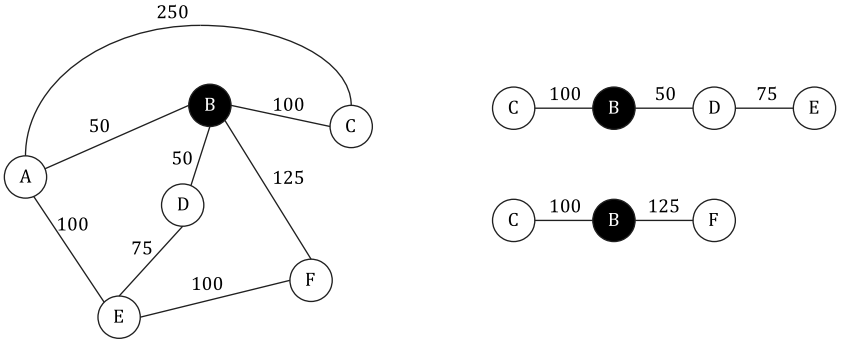

Fig. 2. Example of the optimal solution for network depicted in Fig. 1 that contains a single regenerator in node B (highlighted in black).

Fig. 2 shows the optimal solution for this network, which consists in deploying a regenerator in B. The paths depicted in the right part of the figure are those which were not feasible in the original graph, since they exceeded the maximum allowed distance of 200. However, the deployment of a regenerator in B restores the signal at that node, allowing it to travel a distance of 200 units starting from B. Considering the path that connects C and F, with the regenerator deployed at B, the maximum distance that a signal must travel without reaching a regenerator is 125, resulting from the edges B-D and D-E. This distance is lower than $d_{\max}$, becoming C-B-D-E a feasible path. The same evaluation can be performed for the path that connects nodes C and F.

In this paper, we propose an Iterated Greedy metaheuristic algorithm [4] for solving the RLP. The initial solution is generated following a Greedy Randomized Adaptive Search Procedure (GRASP) [5]. Then, the solution is partially destructed and later reconstructed, combining diversification in the destruction phase with intensification in the reconstruction stage. The reconstructed solution is then post-processed with a method responsible for removing the redundant regenerators. Since the generation of the initial solution is randomized, the process generates several initial solutions, resulting in a multi-start procedure [6], reporting the best solution found during the search.

The remaining of this paper is structured as follows: Section 2 performs a complete review of the RLP literature, Section 3 formally defines the considered problem, Section 4 thoroughly describes the algorithmic approach for the RLP, Section 5 shows the computational experiments performed to analyze the proposed algorithms and compare them with the best previous methods in the state of the art. Finally, Section 6 provides some conclusions derived from the research and presents some ideas for future work on the RLP.

2. Literature review

The RLP arises for the first time in the context of traffic engineering with restoration [1]. In the same year, a multiprotocol label switching over wave division multiplexing problem was proposed, with a constraint that forbids paths between two components larger than a pre-established threshold [2]. The optimization problem was formally defined in [3,7], where authors showed the similarities with the Steiner Arborescence Problem with a unit degree constraint on the root node, proposing a branch-and-cut procedure and several heuristics. The RLP has also equivalences with two other related problems: the Maximum Leaf Spanning Tree Problem (MLSTP) [8] and the Minimum Connected Dominating Set Problem (MCDSP) [9]. The first optimization problem consists in finding the spanning tree of an undirected graph with the maximum number of leaves. Specifically, Chen et al. [7] demonstrated that a solution of the RLP is equivalent to finding a spanning tree in the communication graph (see Section 3 for further details). Therefore, there is an isomorphism between minimizing the number of regenerators in a network and maximizing the number of leafs of a spanning tree of the communication graph. The MLSTP has been tackled from a heuristic perspective by considering a parallel variant of the Variable Neighborhood Search metaheuristic [10].

Similarly, the equivalence between RLP and MCDSP is described in [11]. This optimization problem consists in finding the connected dominating set of minimum cardinality (a subset of vertices of a graph is a dominating set if each edge of the graph has, at least, one endpoint in it). Notice that given an optimal connected dominating set for the MCDSP, we can construct a spanning tree over it resulting in an optimal solution for the MLSTP and vice versa. The MCDSP was also proved to be $\mathcal{NP}$-hard in [12], and several approximation algorithms have been proposed [9,13,14].

A generalized version of the RLP, called Generalized Regenerator Location Problem (GRLP) was recently proposed by Chen

et al. [15]. The GRLP has also been proven to be $\mathcal{NP}$-hard by Chen et al. [7] and Flamini et al. [16]. In this problem, not all nodes can host a regenerator and, also, not all nodes must be connected. Specifically, the problem divides the set of nodes into two disjoint sets: one representing the candidate locations to deploy a regenerator and another one representing the set of terminal nodes that must be able to communicate to each other. Quintana et al. [17] recently proposed a heuristic procedure for solving this problem, becoming the state of the art for the GRLP.

The RLP has been proven to be $\mathcal{NP}$-hard [7,16]. It has been approached by considering both exact and heuristic procedures. From an exact perspective, the RLP is studied by considering flow-based compact formulations and cut formulations, which are based on exploring the equivalence between the RLP and the hub covering location problem with the "closeness" criterion in the closure graph. See [18] for further details. We refer the reader to [19] for an extensive survey of exact algorithms for RLP.

First elaborated heuristics (constructive procedures coupled with local search methods) were presented in [20], where a benchmark of 320 instances of randomly generated networks was also introduced. It is worth mentioning that this set has become the standard testbed for comparing algorithms in the context of RLP. These results were later improved in [21], where a Greedy Randomized Adaptive Search Procedure (GRASP) [5] was proposed, as well as a Biased Random-Key Genetic Algorithm (BRKGA) [22]. In that paper the authors also reported an error in the description of the local search proposed in [20]. In [23] several constructive procedures were presented, where the authors explored the equivalence of the RLP with the MLSTP. These results were further improved in [11], by proposing a tabu search algorithm [24]. This method is referred to as tabu search based iterated metaheuristic (TSIH) in [11]. Experimental results showed that TSIH outperformed all state-of-the-art methods. In particular, it was able to find optimal solutions in 368 instances (out of 480). The authors additionally introduce a set of 150 new large and challenging instances. Considering the equivalence among RLP, MLSTP, and MCDSP, the TSIH method is also tested over 41 MLSTP and 220 MCDSP instances (particularized for 1-1-DSP) respectively, finding the optimal solutions in 37 (out of 41 MLSTP instances) and 218 (out of 220 MCDSP instances).

Summarizing, TSIH emerges as the most competitive method identified in the related literature, obtaining better results than previous approaches in less computing time. Therefore, we select this method to conduct the comparison with the proposed Iterated Greedy procedure.

3. Problem statement

A network is usually represented as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each edge $(u, v) \in E$ has a weight $d_{uv} \in \mathbb{R}$, with $d_{uv} \geq 0$, representing the distance between nodes u and v, with $u, v \in V$. Considering the RLP, a signal can traverse a maximum distance of $d_{\max}$ without any quality impact, where $d_{\max}$ is a problem constraint.

Chen et al. [7] proposed a transformation of the original graph derived from the network into the so-called communication graph. Given the original weighted graph $G = (V, E)$, the transformation starts by deleting all those edges whose weight (distance between its endpoints) is larger than $d_{\max}$, since those edges cannot be used in none of the possible paths. After that, a new edge is added between each pair of nodes whose shortest path has a length smaller or equal than $d_{\max}$ (i.e., those that are feasible paths), since the communication is allowed between them. At this point, the distance information of the edges is not relevant and can be deleted, since there is an edge between each pair of nodes that can communicate without deteriorating the signal. Finally, if the resulting graph, $M = (V, E')$, is complete, then every node can communicate with the remaining ones without deteriorating the signal, so regenerators are not needed in the network. Otherwise, one or more regenerators must be deployed in order to maintain a high quality connection between every pair of nodes. Notice that if the resulting graph M is not connected, then the problem is not feasible, since it is impossible to connect two or more pair of nodes.

Fig. 3 presents the construction of the communication graph derived from the network depicted in Fig. 1. Specifically, Fig. 3.a presents the original weighted network. Then, in Fig. 3.b all edges with distance larger than $d_{\max} = 200$ have been removed (i.e., edge A-C). In Fig. 3.c a new edge is added between each pair of nodes whose shortest path is lower than $d_{\max}$ (i.e., edges A-C, A-D, B-E, C-D, and D-F). At this point, the distance information becomes irrelevant and, as a result, it has been removed in the final communication graph, depicted in Fig. 3.d. Notice that it is necessary to add one or more regenerators since pairs (A,F), (C,E), and (C,F) are still unable to communicate between them (i.e., there is no direct link in the communication graph).

Chen et al. [7] described several interesting properties of the RLP. We will now focus on Lemma 2: *Any minimal solution for the RLP on M can be represented as a spanning tree with regenerators at all internal nodes of the tree*. As an implication of this lemma, they pointed out that the RLP can be solved by finding a spanning tree over the communication graph with the smallest number of internal nodes. As both problems are equivalent, a feasible solution for the RLP can be represented as a spanning tree over the communication graph where all the internal nodes host a regenerator. Furthermore, the solution representation can be simplified since the evaluation of the objective function does not depend on the structure of the resulting tree, but in counting the internal nodes. Therefore, a solution for the RLP consists in finding the minimum connected set of nodes that covers the communication graph M [25]. A set of nodes is a covering for M if and only if each node in M either belongs to the covering or it is adjacent to a node in the covering. We refer the reader to [26] for a detailed description of the equivalence of solving the MLSTP and the Minimum Connected Dominating Set Problem.

It is worth mentioning that all the algorithms presented in this paper represent the solution as a set of nodes that covers the communication graph. In this representation, a regenerator must be deployed in every node belonging to the covering C and, therefore, the objective function value is evaluated as the cardinality of the covering, $|C|$. Finally, the objective of the RLP is to find the smallest cardinality covering among all possible coverings for a given communication graph:

$$RLP(M) = \arg\min_{C \in \mathcal{C}_M} |C| \tag{1}$$

where $\mathcal{C}_M$ is the set of all possible coverings over the communication graph M.

Fig. 4 shows two possible connected sets for the communication graph constructed in Fig. 3.d. The nodes belonging to the connected set are highlighted in black, while the ones covered are highlighted in light gray. White nodes are those not covered by the selected connected set of nodes. Fig. 4.a shows a feasible solution, since all nodes are either in the connected set, or adjacent to a node in it. However, the connected set of Fig. 4.b, conformed by node A, is not a covering and, therefore, not a feasible solution, since node F is not covered (i.e., it neither belongs to the connected set nor is adjacent to a node in the covering). In this case it would be necessary to add a new regenerator to the connected set in order to make the solution feasible.

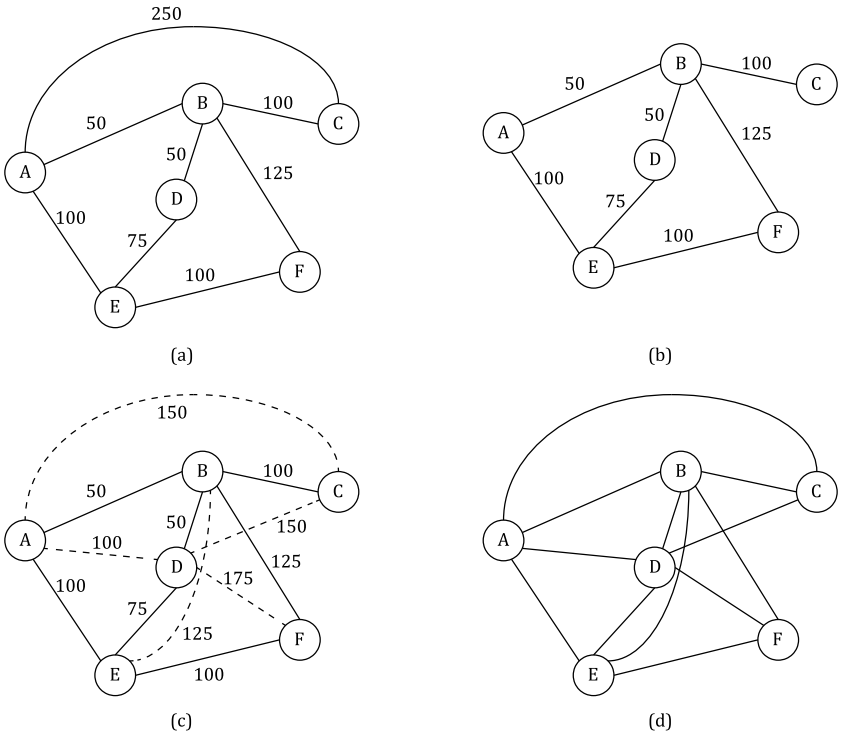

Fig. 3. Construction of the communication graph of the network presented in Fig. 1.

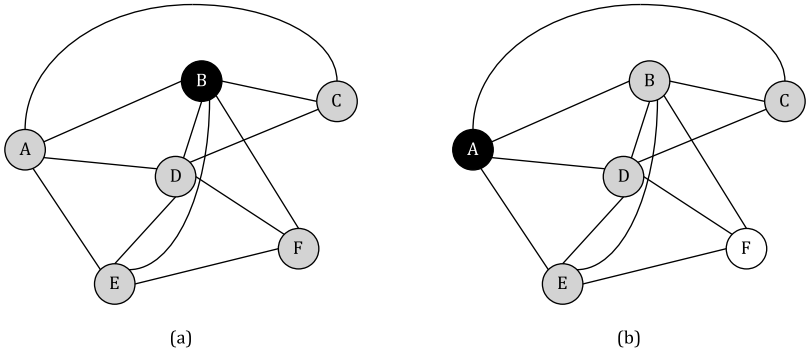

Fig. 4. Example of a connected set that covers the complete graph (a) and another set that does not cover the graph (b).

4. Algorithmic approach

This work tackles the RLP from a metaheuristic point of view. Specifically, we follow the Iterated Greedy (IG) methodology [4, 27]. This metaheuristic framework was originally proposed by Ruiz and Stützle in 2007 for solving a scheduling problem, but it has been successfully applied to different optimization problems in the last years [28,29]. However, the success of IG in the literature is not limited to scheduling problems. Recently, an IG with a Variable Neighborhood Search post-processing for solving a multi-objective waste collection problem with a real application in the city of Málaga (Spain) was proposed [30]. Even in a rather different field such as sentiment classification and analysis IG has proven its validity, providing a feature selection algorithm based on IG for this task [31]. Finally, IG has been also be applied in the context of facility location problem, specifically for locating obnoxious facilities solving the p-median problem for this kind of special facilities [32].

The main idea behind IG metaheuristic is exploring the search space by iteratively applying two main phases: destruction and reconstruction. Algorithm 1 presents the pseudocode of Iterated Greedy.

Algorithm 1 `IteratedGreedy`(S, β)

```
1: while not StoppingCriterion do
2:     S' ← Destruct(S, β)
3:     S'' ← Reconstruct(S')
4:     if f(S) < f(S'') then
5:         S ← S''
6:     end if
7: end while
```

The initial solution S is obtained by a constructive procedure, which is described in detail in Section 4.1. Then, IG enters in the destruction phase (step 2), which basically consists of removing some of the solution components present in S. The number of removed solution components is controlled by parameter β, whose effect in the algorithm results is studied in Section 5. The solution resulting from the destruction phase is usually infeasible, since the destruction phase does not control feasibility during the removal of solution components. Therefore, the reconstruction phase (step 3) is responsible for adding new components to the

solution until it becomes feasible again. Finally, the best solution found during the search is updated if a better solution has been found (steps 4–6). The IG algorithm iterates until a stopping criterion is met (steps 1–7), returning the best solution found during the search.

In this work we propose a Multi-Start Iterated Greedy (MSIG) algorithm. MSIG performs several iterations of the previously described Iterated Greedy algorithm, starting from a different solution in each iteration. The initial solution is generated using the constructive procedure described in Section 4.1. Then, we apply the complete IG algorithm described above, whose destruction and reconstruction phases are described in detail in Sections 4.2 and 4.3, respectively. Each independent IG execution is performed a maximum number of iterations, which is used as a stopping criterion. Then, the procedure starts again from a new initial solution and applies again the destruction and construction phases consecutively. The method stops when a predefined number of initial solutions have been generated, returning the best solution found during the search.

4.1. Constructive procedure

The constructive procedure proposed for the RLP follows a classical Greedy Randomized Adaptive Search Procedure (GRASP) approach [5]. This methodology has led to several successful research, either using it isolated or hybridized with other metaheuristic algorithms, usually with combination methods like Path Relinking [33,34].

The method starts by selecting at random the first node to host a regenerator. Then, it creates a candidate list (CL) with all the nodes adjacent to the first node selected. The rationale behind creating the CL with the adjacent nodes instead of using all possible nodes is to maintain the connectivity in the solution under construction. This idea allows us to save the computing effort of checking, in every step, that the solution is still connected, avoiding the generation of infeasible solutions.

Nodes in the CL are ordered by a greedy function that estimates the effect (in terms of number of nodes covered) of deploying a regenerator in each one of them. Specifically, the greedy function used in this work is defined as the number of nodes that will be covered if a regenerator is deployed in a given node. A node is considered covered if it is either a regenerator or it is directly connected to a regenerator. Given the communication graph M and the solution under evaluation S, the greedy function for a given node v is then defined as:

$$g(v, S, M) = |\{u \in N_M(v) : (\{u\} \cup N_M(u)) \cap S = \emptyset\}| \quad (2)$$

where $N_M(v)$, and symmetrically $N_M(u)$, represent the adjacent nodes to v and u, respectively, in the communication graph M.

The function $g(v, S, M)$ calculates how many new nodes we are covering by selecting node v as a regenerator. Let us illustrate how the greedy function works by considering the communication graph M depicted in Fig. 4. If we consider the vertex A, and an empty initial solution $S = \{\}$, the greedy function $g(A, \{\}, M)$ is equal to 5. Similarly, $g(F, \{A\}, M)$ would be 1, as the other nodes would already be covered.

The Restricted Candidate List (RCL) is created with the most promising candidates to host a regenerator. In particular, given the CL sorted in descending order with respect to the greedy function, the RCL is conformed by those candidate nodes whose greedy function value is larger than or equal to a threshold μ, which is evaluated as:

$$\mu = g_{\max} - \alpha \cdot (g_{\max} - g_{\min}) \quad (3)$$

The value of parameter α in the constructive procedure is in the range [0, 1]. Notice that when α receives the value 0, the constructive procedure becomes purely greedy, and only the nodes with the highest greedy function value can be selected. On the other hand, when $\alpha = 1$, the constructive procedure is entirely random, and any node in the CL can be selected (i.e., RCL = CL). Section 5 performs a deeper analysis on the best value for this parameter.

Once the RCL has been constructed with the most promising nodes, the next node to host a regenerator is selected at random from it. This random selection is performed in order to generate different and diverse solutions in each construction, due to the multi-start nature of our IG algorithm. Having selected the next node, it is added to the solution as a regenerator, and the CL is updated. This update is performed by inserting in the CL those nodes that are adjacent to the selected one that neither have been added to the solution nor belong to the CL yet. The procedure continues adding nodes to the solution until the set of connected nodes in it becomes a covering (i.e., a feasible solution is obtained).

It is worth mentioning that it is not necessary to check the connectivity constraint at every step, since the candidate list is always updated with nodes adjacent to at least one node in the solution, which reduces the computational effort of the constructive procedure.

4.2. Destruction phase

The destruction phase proposed for the Iterated Greedy algorithm is intended to diversify the search by randomly removing some regenerators deployed in a feasible solution. Considering that the covering in a feasible solution is always minimum, the removal of some node of a given solution will eventually lead to infeasible solutions. The infeasibility of a solution after performing the destruction phase can be due to two different reasons: the solution is not a covering any more, or the nodes in the solution are not connected.

With the aim of reducing the computing time required by the algorithm, we only allow the destruction phase to remove those nodes that may produce a solution that is not a covering but it is always a connected set of nodes. Therefore, it is not necessary to execute a connectivity detection algorithm in every step of the search.

The nodes whose removal breaks the connected set into two or more connected components are called articulation points [35]. The most efficient algorithm to detect articulation points in a graph has $O(|V| + |E|)$ complexity. It was originally proposed by Tarjan for finding biconnected components in undirected graphs, and strongly connected components in directed graphs [36]. We use this algorithm, based on performing a depth first search over the graph, in order to identify which are the articulation points of a given connected set of nodes.

As it was aforementioned, the magnitude of the destruction phase is determined by the parameter β. Specifically, this method consists of randomly removing $\beta \cdot |S|$ regenerators. Notice that only those regenerators that are not articulation points are considered for its removal from the solution to maintain it connected. Therefore, if the number of articulation points in the solution under destruction is larger than $(1-\beta)\cdot|S|$, it will not be possible to remove $\beta \cdot |S|$ nodes, resulting in the removal of a smaller number of nodes.

Fig. 5.a illustrates a feasible solution with regenerators at nodes B, D, and E. Additionally, Figs. 5.b and 5.c show the solution resulting from the removal of regenerators in nodes D and E, respectively.

As it was aforementioned, the destruction phase does not allow to remove a node that produces a disconnected set in the solution. Therefore, the removal of the regenerator in node D (Fig. 5.b) would not be considered in the procedure, because it is

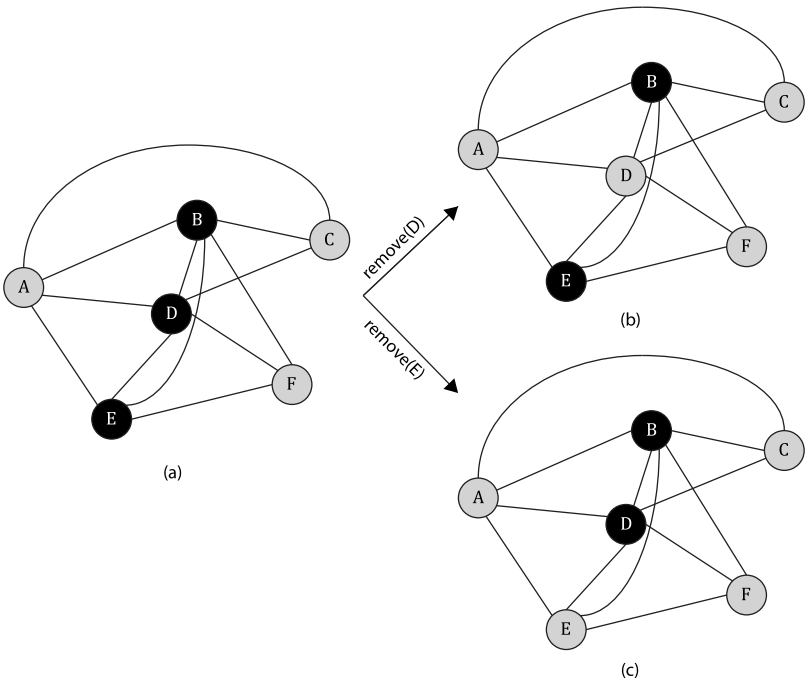

Fig. 5. (a) Example solution before the destruction phase and solutions resulting from the removal of regenerators (b) D and (c) E. Notice that solution in (b) does not maintain connectivity.

an articulation point that splits the solution into two connected sets: {B} and {E}. On the other hand, the removal of node E produces a solution which maintains the connectivity, so it can be considered in the destruction phase of solution depicted in Fig. 5.a.

Notice that, although the method only allows the removal of those nodes that result in connected solutions, this phase can produce solutions that are not feasible, since the remaining regenerators may not form a covering over the communication graph. In this case, the reconstruction phase is responsible for completing the covering and generating a feasible solution.

4.3. Reconstruction phase

The reconstruction phase is devoted to intensifying the search, recovering the feasibility of the solution derived from the destruction phase. Specifically, this phase iteratively adds new regenerators to the solution, stopping when the incumbent solution becomes a covering for the communication graph, making it feasible.

The reconstruction phase follows a greedy criterion in order to minimize the number of regenerators needed to form a feasible solution, in contrast to the destruction phase, where the regenerators were selected at random. The selected greedy criterion for adding new regenerators is the one presented in the constructive procedure (see Section 4.1). The rationale behind using the same greedy criterion is due to the fact that trying to maximize the number of nodes covered in each iteration will minimize the number of regenerators needed, which will eventually lead to better solutions.

Using the same greedy criterion would easily result in the solution used as an input in the destruction phase. In order to avoid this behavior and increase the portion of the solution space explored, the regenerators removed at the destruction phase are not considered in the first step. Instead, the method firstly considers adding as a regenerator those nodes that do not belong to the original solution. Finally, if it is not possible to reconstruct a feasible solution without using the nodes removed in the destruction phase, they are considered as regenerators.

Notice that, in order to maintain the connectivity of the covering and thus reduce the computational effort of the phase, the candidate nodes to host a regenerator are initially those which are adjacent to the ones already in the solution. In each step, nodes adjacent to the selected one are included in the candidate list to be considered in further iterations. This criterion ensures that the inclusion of any node of the candidate list as a regenerator in the solution maintains it as a connected set of nodes. The method stops once the solution conforms a covering, since any additional regenerator would only deteriorate the quality of the solution, increasing the cardinality of the connected dominating set unnecessarily.

Fig. 6 shows a possible reconstruction from an initial solution with a single regenerator located in F (Fig. 6.a). Notice that the set of nodes covered by this solution is {B, D, E, F}, since those nodes are the ones either belonging to the solution (i.e., node F) or adjacent to a node in the solution (i.e., nodes B, D, and E). Therefore, it does not conform a feasible solution, since nodes A and C are not covered yet.

Then, the list of candidate nodes to be added to the solution in the next reconstruction step consists of those nodes that are adjacent to node F, which is the one already in the solution (i.e., nodes B, D, and E). Analyzing the nodes covered by each candidate, we can see that both nodes B and E would cover nodes A and C if they hosted a regenerator, while node E would only cover node A. Therefore, it is interesting to select one node among those with the largest number of nodes covered (i.e., nodes B or D), since they would allow us to find a feasible solution with less nodes in the covering. Ties are broken at random, so in this case node D is selected to generate the solution depicted in Fig. 6.b, obtaining a feasible solution.

4.4. Post-processing

The reconstruction phase described in Section 4.3 can eventually lead to solutions with more regenerators than necessary, as can be seen in Fig. 6.b. Specifically, regenerator deployed in node F is not necessary, since all its adjacent nodes are already

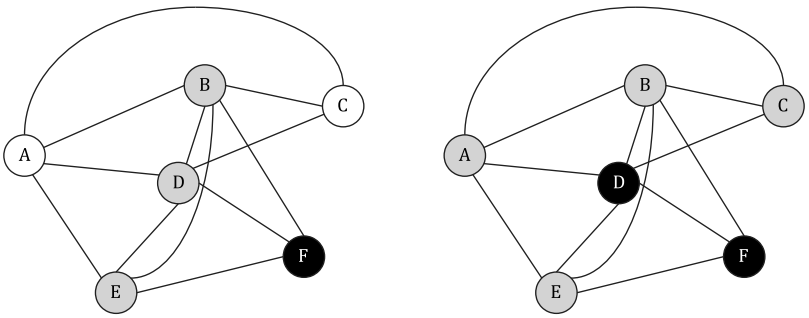

Fig. 6. Reconstruction of solution $S = \{F\}$ by adding a single regenerator in node D.

covered by node D. This result is mainly due to the insertion order of regenerators in the solution, since a regenerator in F is only needed if there is no previous regenerator in D.

We then propose a post-processing phase in order to remove all those redundant regenerators, thus improving the quality of the solutions. The post-processing method is based on verifying, for each regenerator r, that both r and its adjacent nodes are covered only by r itself. If so, the regenerator in r is strictly necessary. Otherwise, all nodes covered by r are also covered by other regenerators, so r is a candidate to be removed.

Finally, a candidate can be effectively removed if and only if its removal maintains the connectivity of the set of regenerators. In particular, a regenerator can only be removed if it is not an articulation point of the regenerators set. This method is executed after the reconstruction of a solution in each step of the Iterated Greedy algorithm.

The complexity of verifying the covering of each node in a given solution is $O(|V| \cdot |E|)$ since, for each node, it is necessary to check all its adjacent nodes. In order to reduce this complexity, we propose a map-based data structure which keeps information about the covering of each node. Specifically, the map stores, for each node of the communication graph, a key–value pair, where the key is a node and the value is a set with all the nodes covering it. Notice that using a set allows us to have constant complexity when checking if a node is covered by a particular regenerator.

5. Computational experiments

This Section reports and analyzes the computational experiments performed for validating the effectiveness and efficiency of the Iterated Greedy algorithm proposed in this work. All the algorithms proposed in this work have been implemented in Java SE 11 and all the experiments have been conducted on a workstation with 16 vCPUs (2.7 GHz) and 8 GB RAM.

In this paper, we consider several benchmark instances previously used in the related literature. In order to facilitate future comparisons, all of them are publicly available at https://grafo.etsii.urjc.es/rlp/. Additionally, we also report individual results per instances and the corresponding CPU time.

Set1: This set contains 41 instances introduced in [37]. These instances were generated as follows. First, a Hamiltonian path is constructed by determining a random permutation for n vertices and linking them using $n - 1$ edges, which ensures its connectivity. Then, edges are randomly introduced in the graph (by considering a uniform distribution) until reaching a defined density (denoted with ρ).

Set2: This set is introduced in [11] and contains 200 instances devised for the 1-1-DSP problem. Each instance is constructed as indicated in [38]. Specifically, nodes are randomly located on a 1000×1000 square area and the transmission range of all sensors is equal to 350. For each network size, 10 instances were generated.

Set3: This set, originally introduced in [7], contains 200 instances. Each one is parametrized by: n, the number of nodes (from 40 to 100); and p, the percentage of non-connected nodes (from 10% to 90%). For each pair n and p, 10 randomly generated networks and constructed.

Set4: This set was also introduced in [7] and contains 200 instances constructed with the same generator. The number of nodes (n) and the percentage of non-connected nodes (p) ranges from 200 to 500 and from 10% to 90%, respectively. As in Set1, there are 10 instances for each pair n and p.

Set5: This set was introduced in [11] and contains 150 instances constructed with the same generator than Set3 and Set4. In this benchmark, the number of nodes (n) and the percentage of non-connected nodes (p) ranges from 600 to 1000 and from 10% to 90%, respectively. As in Set3 and Set4, there are 10 instances for each pair n and p.

Set6: This set is introduced in this work to provide a new challenging benchmark for future comparisons. It contains 250 instances constructed with the NetworkX library [39], using the fast GNP Random Graph generator. In this benchmark, the number of nodes (n) and the percentage of non-connected nodes (p) ranges from 1200 to 2000 and from 75% to 95%, respectively. Specifically, as in Set3, Set4 and Set5, there are 10 instances for each pair (n, p) with $n \in \{1200, 1400, 1600, 1800, 2000\}$ and $p \in \{75\%, 80\%, 85\%, 90\%, 95\%\}$.

The experiments performed are divided into a preliminary experimentation and the final one. The preliminary experimentation is intended to select the best parameters for both the constructive procedure and the Iterated Greedy algorithm. In these experiments, we consider a set of 41 representative instances (approximately 8% of the full set of instances) extracted from Set3 and Set4, in order to avoid overtraining. On the other hand, the final experimentation is devoted to comparing the results obtained by the best configuration with the best previous method found in the state of the art for the RLP [11]. This method is referred to as tabu search based iterated metaheuristic (TSIH).

5.1. Preliminary experimentation

We report the same metrics for all the preliminary experiments. Specifically, the average number of regenerators needed, NR; the average computing time in seconds, Time (s); the average percentage deviation with respect to the best solution found in the experiment, Dev(%); and the number of times that a method matches the best solution found in the experiment, #Best.

The first experiment is devoted to choosing the best value for parameter α of the constructive method. It controls the balance

Table 1
Comparison of effect of parameter α in the constructive method proposed. Best values are highlighted with bold fonts.

α	NR	Time (s)	Dev(%)	#Best
RND	5.02	0.92	2.43	33
0.25	6.34	1.43	20.68	19
0.50	5.68	1.21	14.14	21
0.75	**4.83**	**0.78**	**1.42**	**39**

Table 2
Comparison of the effect of parameters $\%_d$ and IG_{it} in the Iterated Greedy algorithm. Best values are highlighted with bold fonts.

β	*it*	NR	Time (s)	Dev (%)	#Best
	10	4.73	4.88	4.54	32
0.1	25	4.68	4.68	3.79	34
	50	4.54	5.59	0.22	40
	10	4.66	10.48	2.20	35
0.25	25	4.63	10.33	1.76	36
	50	**4.54**	**13.01**	**0.16**	**40**
	10	4.61	20.10	1.67	37
0.50	25	4.61	20.98	1.45	37
	50	4.54	24.52	0.16	40

of the greediness/randomness of the corresponding construction, in such a way that the value of 0 corresponds to a totally greedy algorithm while the value of 1 corresponds to a completely random method. In this experiment, we consider values $\alpha = \{0.25, 0.50, 0.75\}$ and, additionally, we test a variant (*RND*) in which the value of α is selected at random for each iteration, i.e., a different value of α is obtained at random from a uniform distribution in range [0,1] for each construction. As it is customary in GRASP methodology, the constructed procedure is executed for 100 independent iterations for each instance, returning the best solution found. Table 1 shows the results of this experiment, averaging them over the subset of 41 instances.

We can clearly see that all of them present similar values of computing time. However, if we analyze the average deviation and the number of best solutions found, $\alpha = 0.75$ emerges as the best value (highlighted with bold font in Table 1). Furthermore, the average number of regenerators needed is smaller when compared with the others. Therefore, we use $\alpha = 0.75$ for the remaining experimentation.

The next experiment is intended to obtain the best combination of the Iterated Greedy algorithm parameters: the percentage of destruction and the number of iterations. Specifically, for the percentage of destruction, β, we consider the following values: $\beta = \{10\%, 25\%, 50\%\}$, while the number of iterations, *it*, is selected among $it = \{10, 25, 50\}$. Table 2 presents the comparison among different values for β and *it* parameters. It is worth mentioning that the results are obtained when executing the algorithm for 100 different solutions generated with the constructive method previously described, following the multi-start approach described in Section 4.

The first conclusion that can be extracted from the results is that, as expected, the larger the value of *it*, the larger the computing time. Analyzing the average deviation and number of best solutions found, the best results are obtained when performing 50 destruction–reconstruction phases of the IG algorithm.

It is worth mentioning that an increment in the percentage of solution destructed does not lead to better results, but to a considerably larger computing time. Specifically, best results are obtained when considering a destruction of 25%, obtaining the same results than $\beta = 50\%$ but requiring half of the computing time. Therefore, we select $it = 50$ and $\beta = 0.25$ as the best parameters for the Iterated Greedy algorithm (highlighted in bold in Table 2).

5.2. Final comparison

The goal of the final experiment is to validate the results of the proposed Multi-Start Iterated Greedy (MSIG) algorithm when compared with the current state-of-the-art procedures in the context of RLP. On the one hand, we consider the Integer Linear Programming (ILP) model described in [40]. The model has been implemented using the general-purpose solver Gurobi 9.1. We have limited the execution time of ILP to 1800 s since heuristic procedures are quite fast algorithms (i.e., the total execution time is below a dozen of seconds). As a consequence, if the reported time of ILP is shorter than the considered time horizon, Gurobi guarantees the optimality of the solutions found. Symmetrically, an execution time of 1800 s (marked with an asterisk) means that Gurobi is not able to certify the optimality of the solution in that time.

On the other hand, we include in the comparison the best previous heuristic identified in the related literature. This procedure is referred to as TSIH [11]. It generates an initial solution using a greedy constructive procedure which considers the degree of the nodes to generate a solution based on a spanning-tree. After that, a regenerator reduction method is applied which consists in a local search that tries to replace every pair of regenerators with a single one. Finally, the solution obtained is improved with a tabu search metaheuristic.

It is worth mentioning that these experiments are conducted over the whole set of instances. Specifically, we first consider Set1 and Set2 (see [11,37] for further details) originally designed for MLSTP and 1-1-DSP, respectively. As was aforementioned, TSIH finds the optima in almost all instances in very reduced computing times. Similarly, our algorithm also reaches the optimal solution in those instances. Therefore, for the sake of brevity, we omit them. Indeed, we suggest to not include them in future comparisons since they could be considered as "easy to solve".

Table 3 shows the performance of the three compared algorithms over Set3 (small-size instances). Notice that, to ease further comparisons, we report the same metrics than those used in previous works. Specifically, we show for each method the average number of regenerators, NR. Then, for the ILP (executed in our own computer) we report the total CPU time, Time (s); for the TSIH, we indicate the required computing time to reach the best solution, TTB (s) (the authors do not report the total computing time [11]); finally, for our method, we show both, TTB(s) and Time (s). Notice that Gurubi uses the 16 CPU cores available in our computer, while TSIH and MSIG are executed in a single core.

As we can observe in this experiment, ILP is able to find the optimal values in all instances in very reduced computing times. Our method is able to find the optima in all subsets except in those with $p = 90$. Notice that in these subsets, MSIG matches the optima in 24 out of 40 instances (see detailed results in https://grafo.etsii.urjc.es/rlp/). Therefore, in the whole Set3 we found 184 optima out of 200. Regarding the performance of TSIH, we can conclude that this method does not obtain strictly better results than MSIG in any of the instances included in Set3. In fact, MSIG outperforms TSIH in four subsets of instances: $(n = 60, p = 70)$, $(n = 80, p = 90)$, $(n = 100, p = 70)$, and $(n = 100, p = 90)$.

As can be derived from the table, this set can be also considered as "easy to solve" since both heuristic methods are able to obtain the optima in almost all instances by spending negligible CPU time.

We now present in Table 4 the comparison in the performance of the three algorithms when considering Set4 (medium-size instances). These instances are a real challenge for the exact algorithm mainly due to their size and percentage of non-connected nodes (sparsity). These results demonstrate that these instances

Table 3
Comparison among ILP, TSIH and MSIG for instance Set3. Best results are highlighted with bold font.

n	p	ILP		TSIH		MSIG		
		NR	Time (s)	NR	TTB (s)	NR	TTB (s)	Time (s)
	10	**1.40**	0.02	**1.40**	0.00	**1.40**	0.00	0.05
	30	**2.00**	0.02	**2.00**	0.00	**2.00**	0.00	0.04
40	50	**3.00**	0.02	**3.00**	0.00	**3.00**	0.00	0.04
	70	**4.30**	0.05	**4.30**	0.00	**4.30**	0.00	0.05
	90	**8.30**	0.01	8.90	0.00	8.90	0.00	0.08
	10	**1.90**	0.04	**1.90**	0.00	**1.90**	0.00	0.04
	30	**2.00**	0.04	**2.00**	0.00	**2.00**	0.00	0.06
60	50	**3.00**	0.08	**3.00**	0.00	**3.00**	0.00	0.08
	70	**4.80**	0.21	4.90	0.00	**4.80**	0.00	0.09
	90	**9.90**	0.04	10.20	0.07	10.20	0.00	0.16
	10	**1.90**	0.07	**1.90**	0.00	**1.90**	0.00	0.07
	30	**2.30**	0.34	**2.30**	0.00	**2.30**	0.01	0.10
80	50	**3.20**	0.28	**3.20**	0.01	**3.20**	0.00	0.11
	70	**5.00**	0.35	**5.00**	0.00	**5.00**	0.00	0.15
	90	**10.80**	0.21	11.20	0.09	11.10	0.01	0.24
	10	**2.00**	0.11	**2.00**	0.00	**2.00**	0.00	0.10
	30	**2.70**	1.05	**2.70**	0.00	**2.70**	0.00	0.15
100	50	**3.90**	1.25	**3.90**	0.00	**3.90**	0.01	0.14
	70	**5.20**	1.41	5.30	0.03	**5.20**	0.01	0.18
	90	**11.50**	0.92	12.10	0.03	11.90	0.02	0.31

Table 4
Comparison among ILP, TSIH and MSIG for instance Set4. Best results are highlighted with bold font. An asterisk indicates that the ILP solver has not been able to certify the optimality in 1800 s.

n	p	ILP		TSIH		MSIG		
		NR	Time (s)	NR	TTB (s)	NR	TTB (s)	Time (s)
	10	**2.00**	0.75	**2.00**	0.00	**2.00**	0.01	0.37
	30	**3.00**	8.98	**3.00**	0.00	**3.00**	0.00	0.38
200	50	**4.00**	12.26	**4.00**	0.01	**4.00**	0.01	0.49
	70	**6.30**	254.01	6.40	0.07	6.70	0.04	0.57
	90	**14.30**	439.69	14.70	0.41	14.90	0.03	0.89
	10	**2.00**	1.92	**2.00**	0.00	**2.00**	0.01	0.66
	30	**3.00**	59.23	**3.00**	0.01	**3.00**	0.01	0.76
300	50	**4.20**	861.55	4.50	1.03	4.50	0.08	0.79
	70	**7.00**	1694.67	7.10	0.11	**7.00**	0.03	0.94
	90	**16.40**	1800.00*	17.40	0.73	17.00	0.22	1.50
	10	**2.00**	5.37	**2.00**	0.00	**2.00**	0.01	1.13
	30	**3.00**	139.96	**3.00**	0.01	**3.00**	0.06	1.25
400	50	5.00	1800.00*	**4.90**	0.07	**4.90**	0.13	1.23
	70	8.00	1800.00*	8.00	0.32	**7.90**	0.12	1.42
	90	**18.10**	1800.00*	19.00	3.05	18.20	0.59	2.25
	10	**2.00**	11.52	**2.00**	0.00	**2.00**	0.02	2.10
	30	**3.00**	272.08	**3.00**	0.03	**3.00**	0.30	1.94
500	50	**5.00**	1800.00*	**5.00**	0.16	**5.00**	0.02	2.00
	70	**8.00**	1800.00*	8.10	0.60	**8.00**	0.06	2.31
	90	20.10	1800.00*	20.30	4.72	**19.90**	0.31	3.08

are "harder to solve" than those considered so far. Specifically, the ILP approach is only able to prove the optimality in the smallest and densest instances (see detailed results in https://grafo.etsii.urjc.es/rlp/). For the heuristic procedures, results shown in this table are in line with those described in Table 3. In particular, the MSIG is consistently better than TSIH. In addition, our method seems to be even better when considering the most challenging instances (i.e., large values of both, n and p). Furthermore, the time required to reach the best solution is smaller than a second for MSIG in all cases, being a couple of seconds for TSIH in the most demanding instances.

Table 5 shows the results over Set5 (large-size instances), where the number of nodes ranges from 600 to 1000. Analyzing these results, we can observe that ILP is only able to find the optima in the densest subsets (with $p = 10$). In addition, for the largest instances ($n = 1000$), Gurobi is unable even to store the mathematical model in the memory of our computer (marked with '−'). When considering the heuristic procedures,

Table 5
Comparison among ILP, TSIH and MSIG for instance Set 5. Best results are highlighted with bold font. An asterisk indicates that the ILP solver has not been able to certify the optimality in 1800 s. A hyphen indicates that the mathematical model does not fit in memory, and therefore cannot be solved.

n	p	ILP		TSIH		MSIG		
		NR	Time (s)	NR	TTB (s)	NR	TTB (s)	Time (s)
	10	**2.00**	15.83	**2.00**	0.05	**2.00**	0.06	3.79
	30	**3.50**	1219.58	**3.50**	55.83	**3.50**	0.59	3.33
600	50	**5.00**	1800.00*	**5.00**	0.26	**5.00**	0.09	3.15
	70	8.90	1800.00*	8.70	35.14	**8.10**	0.96	3.07
	90	21.00	1800.00*	21.20	7.76	**20.90**	0.32	4.38
	10	**2.00**	67.01	**2.00**	0.01	**2.00**	0.16	7.72
	30	**4.00**	1800.00*	**4.00**	0.19	**4.00**	0.07	6.48
800	50	5.80	1800.00*	5.90	0.87	**5.60**	1.20	5.33
	70	**9.00**	1800.00*	9.10	2.65	**9.00**	0.10	5.29
	90	23.10	1800.00*	23.20	5.68	**22.10**	3.14	6.45
	10	–	–	**2.00**	0.02	**2.00**	1.54	16.52
	30	–	–	**4.00**	0.40	**4.00**	0.17	13.58
1000	50	–	–	**6.00**	1.56	**6.00**	0.13	10.02
	70	–	–	9.90	6.81	**9.60**	0.72	7.58
	90	–	–	24.40	10.46	**23.80**	0.57	9.01

MSIG emerges as the best algorithm for the RLP, reducing the number of required regenerators in a considerably small computing time. It is worth mentioning that, in some cases, MSIG is able to reach the best solution in a hundredth of the time required by TSIH. Even more, in some cases, the average total time of MSIG is even faster than their time to best.

We perform the well-known non-parametric statistical test for pairwise comparisons, in order to validate these results. Specifically, we use the Wilcoxon test [41], which answers the question: do the solutions generated by both MSIG and TSIH represent two different populations? This test has been applied independently to the results obtained in small, medium, and large sets of instances. On the one hand, the p-value for the Wilcoxon test results in 0.205 and 0.059 for Set3 and Set4, respectively, indicating that there are not statistically significant differences between MSIG and TSIH. On the other hand, when applying the test over the set of challenging 150 large instances, the p-value is smaller than 0.05, indicating that the proposed algorithm is significantly better than the previous one.

We finally propose in this paper a new set of much harder instances than those proposed in the related literature so far (Set6). This new set can be publicly downloaded from https://grafo.etsii.urjc.es/rlp/. These instances are notably sparse (with p ranging form 0.95 to 0.75) which become a real challenge for modern heuristics. We report in Table 6 the average number of regenerators together with the time to best and the total time for MSIG.

6. Conclusions

In this paper, we have presented an algorithm based on the Iterated Greedy framework to tackle the Regenerator Location Problem from a heuristic perspective. The performance of the proposed algorithm is compared throughout extensive computational experiments with the best previous method in the state of the art, which corresponds to a Tabu Search algorithm.

The proposed Iterated Greedy (IG) method is based on iteratively destructing and reconstructing a solution until reaching a stopping criterion. We present a destruction and reconstruction method for the RLP as well as a post-processing method which improves the reconstructed solution. Furthermore, the IG algorithm is embedded in a multi-start approach, for which we propose a greedy randomized constructive procedure based on GRASP methodology.

Table 6
MSIG results for the new proposed set of instances, summarized by size and density.

n	p	NR	TTB (s)	Time (s)
1200	75	12.00	0.89	74.09
	80	14.00	6.06	60.32
	85	18.00	3.78	40.65
	90	25.00	6.95	24.26
	95	43.40	5.14	18.06
1400	75	12.00	3.01	119.72
	80	14.70	10.99	82.49
	85	18.80	5.99	59.13
	90	26.00	5.90	42.19
	95	45.00	6.62	22.58
1600	75	12.20	45.88	149.57
	80	15.00	8.54	128.54
	85	19.10	23.80	59.38
	90	27.00	3.88	51.14
	95	46.60	3.60	26.50
1800	75	13.00	3.00	141.29
	80	15.50	35.66	103.62
	85	20.00	4.09	98.01
	90	27.70	14.52	61.59
	95	47.90	9.91	34.05
2000	75	13.00	5.73	227.53
	80	16.00	3.46	161.62
	85	20.00	30.76	106.15
	90	28.00	30.64	72.89
	95	48.90	11.48	39.75

The algorithm requires to configure only three parameters: α, which controls the randomness/greediness of the constructive method; β, which represents the percentage of regenerators removed in the destruction phase; and *it*, which indicates the number of destruction–reconstruction phases to be applied.

The intensive computational results presented show the superiority of our proposal, which is supported by Wilcoxon non-parametric statistical test. According to our experimentation, we suggest using only Set5 and Set6 to find significant differences when comparing modern heuristics in future comparisons. The other instances might still be useful for validation purposes. We do believe that our experimental findings, algorithms, and benchmark instances can be useful for the scientific community as a framework to compare both, new exact and heuristic proposals.

Future lines of research should be focused on designing an efficient and effective local search procedure for this problem. The structure and constraints of the RLP make traditional local search procedures not suitable for it, so it would be interesting to evaluate the impact of new local search procedures in the quality of the produced solutions.

CRediT authorship contribution statement

Juan David Quintana: Conceptualization, Methodology, Software implementation, Writing. **Raul Martin-Santamaria:** Conceptualization, Methodology, Software implementation, Writing. **Jesus Sanchez-Oro:** Conceptualization, Methodology, Software implementation, Writing. **Abraham Duarte:** Conceptualization, Methodology, Software implementation, Writing.

Declaration of competing interest

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

Acknowledgments

This research was funded by "Ministerio de Ciencia, Innovación y Universidades, Spain" under grant ref. PGC2018-095322-B-C22, "Comunidad de Madrid, Spain" and "Fondos Estructurales, Spain" of European Union with grant refs. S2018/TCS-4566, Y2018/EMT-5062.

Appendix. Acronyms

- **BRKGA**: Biased Random-Key Genetic Algorithm
- **CL**: Candidate List, a component of the GRASP constructive method.
- **GRASP**: Greedy Randomized Adaptive Search Procedure.
- **GRLP**: Generalized Regenerator Location Problem.
- **IG**: Iterated Greedy, a metaheuristic algorithm.
- **ILP**: Integer Linear Programming, a linear programming model where at least one variable must take a boolean or integer value.
- **MCDSP**: Minimum Connected Dominating Set Problem.
- **MLSTP**: Maximum Leaf Spanning Tree Problem.
- **MSIG**: Multi-Start Iterated Greedy, the approach presented in this paper.
- **NR**: Number of Regenerators.
- **RCL**: Restricted Candidate List, a component of the GRASP constructive method.
- **RLP**: Regenerator Location Problem.
- **TSIH**: Tabu Search Iterated Metaheuristic, as the previous work refers to its algorithm.
- **TTB**: Time To Best, in other words, how much time is required to reach the stated solution.

References

[1] E. Yetginer, E. Karasan, Regenerator placement and traffic engineering with restoration in GMPLS networks, Photonic Netw. Commun. 6 (2) (2003) 139–149.
[2] L. Gouveia, P. Patricio, A.F. de Sousa, R. Valadas, MPLS over WDM network design with packet level QoS constraints based on ILP models, in: IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), Vol. 1, 2003, pp. 576–586.
[3] S. Chen, S. Raghavan, The regenerator location problem, in: Proceedings of the 2007 International Network Optimization Conference (INOC 2007), Spa, Belgium, 2007.
[4] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European J. Oper. Res. 177 (3) (2007) 2033–2049.
[5] T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures, J. Global Optim. 6 (2) (1995) 109–133.
[6] R. Martí, J.M. Moreno-Vega, A. Duarte, Advanced multi-start methods, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Springer US, Boston, MA, 2010, pp. 265–281.
[7] S. Chen, I. Ljubić, S. Raghavan, The regenerator location problem, Networks 55 (3) (2010) 205–220.
[8] H.-I. Lu, R. Ravi, The power of local optimization: Approximation algorithms for maximum-leaf spanning tree, in: Proceedings of the Annual Allerton Conference on Communication Control and Computing, Vol. 30, University of Illinois, 1992, p. 533.
[9] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (4) (1998) 374–387.
[10] J. Sánchez-Oro, B. Menéndez, E. Pardo, A. Duarte, Parallel strategic oscillation: An application to the maximum leaf spanning tree problem, Prog. Artif. Intell. 5 (2) (2016) 121–128.
[11] X. Li, C. Yue, Y. Aneja, S. Chen, Y. Cui, An iterated tabu search metaheuristic for the regenerator location problem, Appl. Soft Comput. 70 (2018) 182–194.
[12] M.R. Gary, D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, 1979.
[13] S. Butenko, X. Cheng, D.-Z. Du, P.M. Pardalos, On the construction of virtual backbone for ad hoc wireless network, in: S. Butenko, R. Murphey, P.M. Pardalos (Eds.), Cooperative Control: Models, Applications and Algorithms, Springer US, Boston, MA, 2003, pp. 43–54.

[14] M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, D.J. Rosenkrantz, Simple heuristics for unit disk graphs, Networks 25 (2) (1995) 59–68.
[15] S. Chen, I. Ljubić, S. Raghavan, The generalized regenerator location problem, INFORMS J. Comput. 27 (2) (2015) 204–220.
[16] M. Flammini, A. Marchetti-Spaccamela, G. Monaco, L. Moscardelli, S. Zaks, On the complexity of the regenerator placement problem in optical networks, IEEE/ACM Trans. Netw. 19 (2) (2011) 498–511.
[17] J. Quintana, J. Sánchez-Oro, A. Duarte, Efficient greedy randomized adaptive search procedure for the generalized regenerator location problem, Int. J. Comput. Intell. Syst. 9 (6) (2016) 1016–1027.
[18] J.F. Campbell, Integer programming formulations of discrete hub location problems, European J. Oper. Res. 72 (2) (1994) 387–405.
[19] X. Li, Y.P. Aneja, Regenerator location problem: Polyhedral study and effective branch-and-cut algorithms, European J. Oper. Res. 257 (1) (2017) 25–40.
[20] S. Chen, I. Ljubić, S. Raghavan, The regenerator location problem, Networks 55 (3) (2010) 205–220.
[21] A. Duarte, R. Martí, M. Resende, R. Silva, Improved heuristics for the regenerator location problem, Int. Trans. Oper. Res. 21 (4) (2014) 541–558.
[22] J.F. Gonçalves, M.G. Resende, Biased random-key genetic algorithms for combinatorial optimization, J. Heuristics 17 (5) (2011) 487–525.
[23] C. Yue, X. Li, K. Wei, S. Lin, Heuristics for the regenerator location problem, in: 2014 International Conference on Computer, Network Security and Communication Engineering, Vol. 7, DEStech Publications, Inc., Shenzhen, China, 2014, pp. 641–647.
[24] F. Glover, Tabu search—Part I, ORSA J. Comput. 1 (3) (1989) 190–206.
[25] L. Simonetti, A. Salles da Cunha, A. Lucena, The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm, in: Network Optimization: 5th International Conference, INOC 2011, Hamburg, Germany, June 13-16, 2011. Proceedings, Springer, Berlin, Heidelberg, 2011, pp. 162–169.
[26] H. Fernau, J. Kneis, D. Kratsch, A. Langer, M. Liedloff, D. Raible, P. Rossmanith, An exact algorithm for the maximum leaf spanning tree problem, Theoret. Comput. Sci. 412 (45) (2011) 6290–6302.
[27] R. Ruiz, T. Stützle, An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives, European J. Oper. Res. 187 (3) (2008) 1143–1159.
[28] J. Dubois-Lacoste, F. Pagnozzi, T. Stützle, An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem, Comput. Oper. Res. 81 (2017) 160–166.
[29] Z. Yuan, A. Fügenschuh, H. Homfeld, P. Balaprakash, T. Stützle, M. Schoch, Iterated greedy algorithms for a real-world cyclic train scheduling problem, in: Hybrid Metaheuristics: 5th International Workshop, HM 2008, MáLaga, Spain, October 8-9, 2008. Proceedings, Springer, Berlin, Heidelberg, 2008, pp. 102–116.
[30] L. Delgado-Antequera, R. Caballero, J. Sánchez-Oro, J. Colmenar, Martí, Iterated greedy with variable neighborhood search for a multiobjective waste collection problem, Expert Syst. Appl. 145 (2020) 113101.
[31] O. Gokalp, E. Tasci, A. Ugur, A novel wrapper feature selection algorithm based on iterated greedy metaheuristic for sentiment classification, Expert Syst. Appl. 146 (2020) 113176.
[32] O. Gokalp, An iterated greedy algorithm for the obnoxious p-median problem, Eng. Appl. Artif. Intell. 92 (2020) 103674.
[33] A. Duarte, J. Sánchez-Oro, M. Resende, F. Glover, R. Martí, Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization, Inform. Sci. 296 (2015) 46–60.
[34] V. Campos, R. Martí, J. Sánchez-Oro, A. Duarte, GRASP with path relinking for the orienteering problem, J. Oper. Res. Soc. 65 (12) (2014) 1800–1813.
[35] L. Tian, A. Bashan, D. Shi, Y. Liu, Articulation points in complex networks, Nature Commun. 8 (1) (2017) 1–9.
[36] R. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (2) (1972) 146–160.
[37] A. Lucena, N. Maculan, L. Simonetti, Reformulations and solution algorithms for the maximum leaf spanning tree problem, Comput. Manag. Sci. 7 (3) (2010) 289–311.
[38] N. Ahn, S. Park, An optimization algorithm for the minimum k-connected m-dominating set problem in wireless sensor networks, Wirel. Netw. 21 (3) (2015) 783–792.
[39] V. Batagelj, U. Brandes, Efficient generation of large random networks, Phys. Rev. E 71 (3) (2005) 036113.
[40] B. Yıldız, O.E. Karaşan, Regenerator location problem and survivable extensions: A hub covering location perspective, Transp. Res. B 71 (2015) 32–55.
[41] E.A. Gehan, A generalized wilcoxon test for comparing arbitrarily singly-censored samples, Biometrika 52 (1–2) (1965) 203–224.