Benchmarking a Wide Spectrum of Metaheuristic Techniques for the Radio Network Design Problem

Silvio P. Mendes, Guillermo Molina, Miguel A. Vega-Rodríguez, Juan A. Gómez-Pulido, Yago Sáez, Gara Miranda, Carlos Segura, Enrique Alba, Pedro Isasi, Coromoto León, and Juan M. Sánchez-Pérez

Abstract—The radio network design (RND) is an NP-hard optimization problem which consists of the maximization of the coverage of a given area while minimizing the base station deployment. Solving RND problems efficiently is relevant to many fields of application and has a direct impact in the engineering, telecommunication, scientific, and industrial areas. Numerous works can be found in the literature dealing with the RND problem, although they all suffer from the same shortfall: a noncomparable efficiency. Therefore, the aim of this paper is twofold: first, to offer a reliable RND comparison base reference in order to cover a wide algorithmic spectrum, and, second, to offer a comprehensible insight into accurate comparisons of efficiency, reliability, and swiftness of the different techniques applied to solve the RND problem. In order to achieve the first aim we propose a canonical RND problem formulation driven by two main directives: technology independence and a normalized comparison criterion. Following this, we have included an exhaustive behavior comparison between 14 different techniques. Finally, this paper indicates algorithmic trends and different patterns that can be observed through this analysis.

Index Terms—Antennae, benchmarking, evolutionary algorithms, metaheuristics, optimization, radio network design (RND).

I. INTRODUCTION

WITH THE fast growth and merging of communication infrastructures and services, the planning and design of wireless networks has become a very complex subject. Despite the attention it has received from the scientific community, this field of research in optimization is still relatively obscure. Moreover, present-day industry expertise is generally based on ad hoc or nonformal approaches. A body of scientific work has been developed around the radio network design (RND) optimization problem, but all the studies suffer from the same shortfall—noncomparable efficiency. RND plays a major role in various engineering, industrial, and scientific applications because its outcome usually directly affects cost, profit, or other heavy-impact business performance metrics. This means that the quality of applied RND approaches has a direct bearing on industry economic plans. The evolution of radio network technology has made this scenario recurrent as a result of successive experimental approaches to optimization which mainly consider the technological aspects of the RND problem instead of its canonical formulation. As a direct consequence, it remains impossible to identify the most effective formal method to tackle an instance of the RND problem.

The main purpose of this paper is to offer a reliable base reference for RND comparison, with foundation-dissimilar approaches in order to cover a high algorithmic spectrum, thus converting it into a valuable edifying tool for potential experimental applications vis-à-vis new or yet-to-come radio network technology.

Our comparison methodology relies on the canonical RND problem formulation and is governed by two main directives: technology independence and a normalized comparison criterion. The technology independence is achieved by neglecting any of the additional technological constraints that would be thrown into the problem, as for instance part of the base station (BS) properties definition (antenna, azimuth, and tilt), path loss models, bandwidth zone prediction, etc. Instead, we consider a theoretical isotropic radiating model, which is mainly used as a reference radiating model. Since RND is a well-known NP-hard problem [1], technological constraints would only raise the combinatorial complexity while the problem’s essence would remain untouched. The normalized comparison criterion is based on a fitness evaluation effort metric (FEEM) since real-world applications would spend most of their computing effort on the evaluation of real-wave-based solutions rather than of the algorithm per se.

Previous partial experiments have been conducted [2]–[8], but this paper combines results and scales up to a real-world-sized problem, in order to accurately compare efficiency, reliability, and swiftness at such an magnitude.

1089-778X/$26.00 © 2009 IEEE
This paper is organized as follows. Section II discusses related work. In Section III, we go on to introduce the RND problem, and in Section IV we present the clock calibration model. In Section V we review the canonical models of the algorithmic approaches, while our main results are presented in Section VI. Section VII concludes the paper and identifies areas for future work.

II. RELATED WORK

The purpose of this section is to describe the related work that is closely associated with the paper’s main subject. The inclusion of this section is vindicated by the absence of a formalized RND benchmark, which makes it impossible to know which is the best RND optimizing formulation. We divide this section into four distinct areas of impact.

A. RND Research Foundations

Despite some resemblances to Calégari’s [9], [10] work on genetic algorithmic (GA) approaches for radio network optimization for mobile systems, developed in the mid-1990s, this field of research actually focuses on the principle of minimization of resources rather than on achieving the total coverage of an area, since in most real-world-problem cases, these latter scenarios are uncommon. Calégari GAs adopted the graph-maximum independent-set search method which attempts to find the largest independent-set in a graph.

Since then, many GAs have been applied with an uncertain degree of success. Additional examples are [7], [8], [11], [12], including several parallel and multiobjective implementations [13]–[16]. Calégari et al. developed his RND dominating set model (still supported by the maximum independent-set search method) based on a hybrid implementation that combines a greedy algorithm with his previous GA development, and presenting it in the form of a framework formally known as STORMS (software tools for the optimization of resources in mobile Systems). This approach had in mind some Universal Mobile Telecommunications System particularities. Several initiatives have been developed on the STORMS platform. Chamaret et al. [18] followed Calégari’s work and tested seven different heuristics on the STORMS framework, employing the maximum independent-set search method.

Nevertheless, several distinct approaches have also been identified. He et al. performed a unique related work [19] that consists in applying a pattern search algorithm called Dividing RECTangles (DIRECT) proposed by Jones et al. [20]. The distinctive feature of this paper is that their algorithmic approach was connected to a parallel 3-D radio propagation ray-tracing module running on a 200-node Beowulf cluster of workstations. A high degree of focus was placed on the 3-D ray-tracing propagation model, based on geometrical optics, when computing BS site power levels.

Isolated analytical and heuristic proposals are also found in this field. Vasquez et al. proposed a Tabu-based heuristic approach for antenna positioning [21] using the quintuplet BS compound (site, antenna, tilt, azimuth, and power). Elkachouchi et al. [22] developed work based on a particle swarm optimization approach and included morphological data in their internal representation matrix.

Finally, RND-directed research work is found wherein a demand-based criterion has primarily been taken into account, i.e., predicting traffic density. This kind of work, while falling outside the scope of our main subject, is relevant because of some proposed novelties at algorithmic level. Tutschku proposes a simple greedy-based heuristic, named SCBPA [23], applied on the maximal coverage location problem [24] with heavy restrictions on predicted traffic density. Ibbetson et al. propose two simple heuristics based on excess traffic redistribution of BS [25], and Fritch et al. propose an approach based on self-organizing sensory neurons implemented via simulated annealing [26].

B. Local Area RND Research

The same principles and requirements determined by the main streaming of wireless LANs (WLAN) also contribute to this line of research, since the problem itself is equivalent. When positioning access points on WLAN design (also commonly referred to as indoor network optimization), the log distance path loss model is an essential requirement for solving these particular types of RND sub-problems. Kamenetsky et al. have proposed some solutions based on an extension of simulated annealing through a pruning scheme to obtain a reasonable initial solution [27]. Bahri et al. based their work on a tabu-search algorithm [28], Aguado-Agelet et al. resorted to a more traditional genetic approach [29], while Fruhwirth et al. used simple constraint-based programming as an optimization technique [30], to name only a few. Much more work focusing on this particular RND problem can be found in [31]–[36].

C. RND Applied Research

Although some works related to RND applied research are found (namely 2G and 3G), few demonstrate optimization techniques, preferring to confine themselves to the proposal of planning methods or procedures [37], [38]. A hierarchical parallel approach for GSM network design has been proposed by Talbi et al. [39]. It stands on Alba’s previous work [7], extending his original proposed GA approaches.

D. RND-Related Commercial Packages

Several RND-related commercial packages are also found, although little evidence of automatic optimization has been reported in any of them. A common feature is their assistance in the designing of a network.

For instance, Mentum Planet [40] provides an optimization software solution for wireless access networks. It offers a range of capabilities that support the evolving role of wireless network planning solutions, such as the convergence of multitechnology networks.

France Telecom’s Research and Development Group [41] manages more than 8400 patents and belongs to the Orange Lab’s worldwide network. Reports refer to network optimization research, although this is not in the public domain.
discretized on a grid, (b) a graph, whose edges link transmitters to the location they cover, and (c) the Bipartite graph representation, where $M$ is the set of potential transmitters and $L$ is the set of potentially covered locations.

Fig. 2. (a) Three potential BS $e_1$, $e_2$, $e_3$ and the associated cells on a discrete surface, (b) a graph set, whose arcs link transmitters to the location they cover, where the dotted lines represent directed hyperedges and vertex coloring has been considered for the irradiation sources, and (c) the hypergraph representation is closely analogous to its discretized surface counterpart.

E. RND-Related Work Concluding Notes

As a concluding remark, the vastness of the related work clearly demonstrated to us that there is a repetitive research cycle each time a new radio-based technology emerges. Such research in turn typically endeavors to adapt previous algorithms without any overriding consideration of research related to previous generation technology. Each of the works examined tackles a specific RND problem based on specific technology-dependent features, and they all use a myriad of optimization approaches. Every work additionally concludes how good and promising the results achieved are, but indeed it is very unlikely that all of the previous works achieve reliable optimal results.

III. RND-Enhanced Problem Formulation Model

The RND optimization problem comprises the maximization of the coverage of a given geographical area while minimizing the BS deployment, hence is an intrinsically multiobjective problem. A BS transmitter is a radio signal transmitting device that irradiates any type of wave model, and the part of the area that is covered by a BS is called a cell. If two or more BS transmitters are close to each other, their cells can overlap, and the locations inside these areas might have different degrees of coverage (for example, one location can be under the influence of two BS transmitters while another can be inside the cell of only one transmitter; in this case the second location has a lower level of intensity of the received signal). Calégari proposes a dominating set model [42], derived from graph theory, that is very similar to the minimum dominating set problem [1]. His approach considers a graph $G = (V, E)$ where $V$ is a set of vertices and $E$ is a set of edges. The dominating set of $G$ is a subset $V' \subseteq V$ defined by

$$\forall u \in V / V', \exists v \in V' \text{ such that } (v, u) \in E.$$  \hspace{1cm} (1)

A dominating set $V'$ is said to be minimum if no other dominating sets have a smaller size (number of vertices).

Afterward, Calégari redefined his initial approach [10] in a way such that $V$ is in fact the union of two sets $M$ and $L$, where $M$ is the set of all possible BS locations, and $L$ is the set of all potentially covered locations, formulating the graph $G = (M \cup L, E)$, where $E$ is a set of edges such that each transmitter location is linked to the locations it covers. Related work is also found based upon the maximal independent-set problem [17]. Dominating sets are closely related to independent-sets such that a maximal independent-set in a graph is an upper bound of the minimal dominating set. However, dominating sets need not be independent. Fig. 1 shows how, according to Calégari, the potentially covered locations are taken from a discretized geographical area.

Since the objective of RND is to search for the minimum subset of transmitters that covers a maximum surface of an area, we are searching for a subset $M' \subseteq M$ such that $[M']$ is minimum and such that $\text{Neighbors}(M', E)$ is maximum, where

$$\text{Neighbors}(M', E) = \{u \in L | \exists v \in M', (u, v) \in E\}.$$  \hspace{1cm} (2)

Nevertheless, this model ignores the fact that the same element can belong to both $M$ and $L$ simultaneously, since $M \subset L$, which will mostly inflict coverage changes at some points. In other words, it can be that a potential transmitter location is also subject to being a covered location, meaning replication in $M$ and $L$ in such a way that it voids Calégari’s bipartite representation shown in Fig. 1(c). Additionally, he also disregards the direction of each $E \in E$.

We propose an alternative model, extended and updated from the one previously given, which is based on a hypergraph $H = (V, E')$ (a set of graphs), where $E$ represents the hyperedges (edges that can contain any subset of vertices of a hypergraph) and $V = M \cup L$. Our hypergraph $H$ is a set system composed of directed monomial graphs $p(x) = x^E$ in such a way that a central vertex is achieved in each one of them. Each central vertex represents an irradiating source ($M'$) and hyperedges are arcs in the form of $E' = (x, y)$, directed from $x$ to $y$ [in this case any number of vertices can belong to $E'$ within the $(x, y)$ boundary]. This formal model tends to be more accurate when compared to the real problem: 1) The bipartite graph is not always possible to achieve (which means that the problem formulation increases its complexity level, although its combinatorial echelon is not affected) and 2) the radiating sources are directed, as expressed by the arcs, even if omnidirectional. Fig. 2 shows how the current extended model can be extracted from a discrete surface.

A fitness function $f$ is required to evaluate the quality of a BS set $H'$. The fitness is described by the ratio of the
square of the cover rate and the number of BS transmitters used (3). It has also been reported as being widely adopted in the telecom industry [10], hence its usage in our experiments. Our BSs employ an omnidirectional isotropic wave model, which is a theoretical point source that radiates uniformly in all directions. The coverage provided by the BSs is recorded on the coverage grid in order to speed up fitness calculations (no graph structures have been employed). Topography has been partially disregarded (wave collision objects or geomorphic data) to generalize the problem

\[ f(x) = \frac{\text{CoverRate}(x)^2}{\text{Number Transmitters Used}(x)} \]  

where

\[ \text{CoverRate}(x) = \frac{100 \times \text{Neighbors}(M', E')}{\text{Neighbors}(M, E)} \]  

The problem we consider reminds of the **unicost set covering problem** (USCP), which is a known NP-hard combinatorial optimization problem [1]. The RND problem differs, however, from the USCP in that the goal is to select a subset of BS that ensures a good coverage of a given area, and not to ensure a total coverage. This emphasizes the principle of minimization of resources rather than achieving the total coverage of an area, since in most real-world-problem cases, these latter scenarios are uncommon. On the other hand, the RND NP-hardness is maintained in magnitude and is as tightly related to the field of combinatorial mathematics as USCP.

**IV. RND Formal Comparison Specification**

The RND formal comparison specification describes the accurate formal process that is used when comparing diverse implementations that aspire to RND optimization.

The FEEM has been conceived to normalize the comparison of heterogeneous RND runtime environment discrepancies, including hardware and software issues (also known as benchmark routines for system clock calibration). After conducting profiling experiments employing an omnidirectional isotropic reference radiation wave model, we defined a clock-calibration FEEM that effectively replaces wall-clock measuring, allowing the disregarding of hardware runtime platforms, chiefly the processor(s) frequency or architecture (like parallelization through core replication). Software issues are also minimized whenever using highly optimized code compilers (like gcc) or intermediate compilers allowed on some interpreting enterprise runtime platforms (like J2EE), although raw wall clock times can remain considerably different.

The radio propagation wave model computing effort is highly correlated with the algorithm’s wall clock. Due to the complexity of real-wave model calculations, the computing effort of the algorithm is irrelevant when compared to the total computing effort. Although this is true in other problems, the RND problem presents an extreme case of such a scenario.

Population-based approaches rely on light heuristics, intrinsically defined by their breeding operators. In these cases, a whole solution can be evaluated at once. Hence one FEEM unitary value can be summed each time a solution is evaluated.

Algorithms relying on heavy-based heuristics and/or incremental fitness evaluations do not always compute a complete solution at a given time. In these cases, the elements that compose the solution are mostly used to partially evaluate the solution. Consequently 1/is FEEM is summed for each compound-based element that is evaluated, where is represents the solution instance size. A single algorithm can use both FEEM profiling in a single run, depending on the type of evaluations employed. Both methods represent a compatible additive metric that can be compared.

In this paper, the use of population-based methods implies a whole objective function evaluation per individual per generation, because incremental computation of the fitness function does not pay off, according to our preliminary experimental data.

**V. Optimization Algorithms Overview**

In this section, models of the optimization algorithms employed throughout this paper are provided. The main objective is to offer an overall picture, including many representative classes, for instance, direct search techniques, random guided search techniques, genetics, several evolutionary strategies, and models that include bio-inspired aspects. A total of 14 significant models are described and further analyzed (nine distinct approaches and five hybrids or variants). Each research group has focused on single implementations in order to reduce variability and achieve the best results. All the approaches presented below have a common stopping criterion defined by 5000000.00 fitness evaluations in compliance with the FEEM definition (Section IV-A).

**A. Simulated Annealing**

Simulated annealing (SA) is a trajectory-based optimization technique. It was first proposed by Kirkpatrick et al. in [43]. SA is a fairly commonly used algorithm that provides good results [44]–[48] and constitutes an interesting method to compare to other optimizing methods.

**Simulated Annealing Pseudocode**

\[ t \leftarrow 0 \]

Initialize \((T, S_0)\)

while not end condition \((t, S_0)\) do

while not cooling condition \((t)\) do

\[ S_n \leftarrow \text{Choose neighbor } (S_n) \]

Evaluate \((S_n, S_n)\)

if Accept \((S_n, S_n, T)\) then

\[ S_n \leftarrow S_n \]

end if

\[ t \leftarrow t + 1 \]

end while

Cooldown\((T)\)

end while

return \(S_n\)

The algorithm works iteratively and keeps a single tentative solution \(S_n\) at any time. At every iteration, a new solution \(S_n\) is generated from the old one \(S_n\), and depending on some
acceptance criterion it might replace it. The acceptance criterion is the true core of the algorithm. It works as follows: both the old $S_o$ and the new $S_n$ solutions have an associated quality value—determined by a fitness function. If the new solution is better than the old one, then it will replace it. If it is worse, there is still some chance that it will replace it. The replacing probability is calculated using the quality difference between the two solutions and a special control parameter $T$ named temperature. An initialization process is employed to select the starting value of the temperature. Once the initial solution has been generated, we select a value of $T$ such that any neighbor of the initial solution will be accepted with probability 0.8. This is achieved by successively trying different temperature values with 100 random neighbors and checking the acceptance criterion until the mentioned constraint is met. The aim here is to guarantee the initial randomness of the search process.

The acceptance criterion ensures a way of escaping local optima by choosing solutions that are actually worse than the previous one with some probability. That probability is calculated using Boltzmann’s distribution function

$$P = \frac{2}{1 + \exp\left(\frac{\text{fitness}(S_n) - \text{fitness}(S_o)}{T}\right)}.$$  \hspace{1cm} (5)

As the iterations go on, the value of the temperature parameter $T$ is progressively reduced following a cooling schedule, thus reducing the probability of choosing worse solutions and increasing the biasing of SA toward good solutions. In this paper, we employ a geometric rule, such that for every $k$ (Markov chain length) iterations the temperature is updated as $T(n+1) = \alpha T(n)$, where $0 < \alpha < 1$ is called the temperature decay. The cooling schedule employed is a standard one. Our interest was to test the SA archetype, therefore the canonical form was chosen for the algorithm’s internal operations.

In this paper we employ binary vector variables and bit-flip mutation to generate new solutions. For the bit-flip procedure, a mutation probability $P$ is selected. Every bit in the bit string is sequentially visited, and then with probability $P$ that bit is flipped. Therefore, for a string of length $L$, $L*P$ bits are flipped on average by the bit-flip mutation. For the encoding a binary vector was employed (each bit uniquely corresponds to a candidate location site; a 1 indicates that a transmitter is placed, a 0 indicates that no transmitter is placed).

### B. CHC

Eshelman's cross generational elitist selection, heterogeneous recombination, and cataclysmic mutation (CHC) is a kind of evolutionary algorithm (EA) surprisingly not used in many studies despite having unique operations usually leading to very efficient and accurate results [49]. Like most EAs, CHC relies on a set of solutions (population, hereafter referred to as $P_a$). The algorithm proceeds iteratively and at the end of each iteration some solutions will be replaced by newly created ones. At every step, a new set of solutions is produced by selecting pairs of solutions from the population (the parents) and recombining them. This selection is made in such a way that individuals that are too similar (Hamming distance below a given threshold) cannot mate each other, and recombination is made using a special procedure known as Half Uniform Crossover. This procedure first copies the common information for both parents into both offspring, then translates half the diverging information from each parent to each of the offsprings. This is done in order to preserve the maximum amount of diversity in the population, as no new diversity is introduced during the iteration (there is no mutation operator). The next population $P_n$ is formed by selecting the best individuals among the old population and the new set of solutions (elitist criterion).

As a result of this, at some point in the execution, population convergence is achieved, so the normal behavior of the algorithm should be to stall on it. The threshold is progressively reduced to encourage the production of new solutions when the population begins to converge. When convergence is finally reached, a special mechanism is used to generate new diversity: the restart mechanism. When restarting, all of the solutions except the very best ones are significantly modified (cataclysmically). This way, the best results of the previous phase of evolution are maintained and the algorithm can proceed again. The algorithm’s general procedure is described in the following pseudocode:

**Eshelman’s CHC Pseudocode**

```plaintext
$t \leftarrow 0$
Initialize ($P_a$, convergence count)
while not ending condition ($t$, $P_a$) do
    Parents $\leftarrow$ Selection parents ($P_a$)
    Offspring $\leftarrow$ HUX (Parents)
    Evaluate (Offspring)
    $P_n$ $\leftarrow$ Elitist selection (Offspring, $P_a$)
    if not modified($P_a$, $P_n$) then
        convergence count $\leftarrow$ convergence count$-1$
        if convergence count $== 0$ then
            $P_n$ $\leftarrow$ Restart($P_a$)
            Initialize(convergence count)
    end if
    end if
    $t \leftarrow t + 1$
    $P_a$ $\leftarrow$ $P_n$
end while
```

We employ bit-flip mutation with a flip probability of 35% for the cataclysmic mutation, 100 individuals, and a crossover rate of 80%. All the parameters were determined by experimental fine-tuning. Binary encoding was employed.

### C. Iterated Local Search

Iterated local search (ILS) is a neighborhood exploration paradigm that was initially introduced by Lourenço et al. [50]. It is a simple and generally applicable metaheuristic which iteratively applies local search (LS) to modifications of the current search point. ILS improves the performance of local searches by allowing them to escape local-optima trapping and...
continue the search for possible better solutions. The success of ILS lies in the biased sampling of this set of local optima.

For the resolution of the RND problem, we have defined specific methods for the generation of the initial solution and for the perturbation of the solutions. The algorithm’s general procedure is described in the following pseudocode:

**Iterated Local Search Pseudocode**

```plaintext
s_0 ← generateInitialSolution()

s ← localSearch(s_0)

S_best ← s;

while not end condition() do

  s' ← perturbation(s, history)

  s'' ← localSearch(s')

  if S_best BetterThan s'' then

    S_best ← s''

  end if

  s ← acceptanceCriterion(s, s'', history)

end while

return S_best
```

At the start of the algorithm, an initial solution is generated. For the generation of the initial solution, we have designed a specific problem-based heuristic. The developed heuristic divides the grid of the terrain to be covered into a set of sub-grids or windows. All windows have size $N \times N$, where $N$ is randomly selected from the values nearest to twice the antennae coverage radius. On each window, a base station transmitter is placed in the most centered location available. In addition, the central position of the windows can be slightly displaced from the grid reference coordinates. Afterward, a local search is performed on this initial solution. Then, the search loop is repeated until some stopping criterion is satisfied. In our case, the end condition() function tests whether the algorithm has performed the total number of evaluations, which is a common criterion for all algorithms. In each cycle, a diversification step is first applied by perturbing $s$ to obtain a new solution $s'$. Intensification is then performed around $s'$ by applying a local search to produce a new solution $s''$. If $s''$ satisfies an acceptance criterion, it replaces $s$ and the next step is carried out from this new solution. The acceptanceCriterion($s, s'', history$) function returns the best solution between $s$ and $s''$ depending on their fitness values.

The designed perturbation mechanism selects a set of deployed transmitters to be removed from the solution and a set of locations in which to include an extra antenna. For perturbing any solution, the number of transmitters to be deleted and also the ones to be inserted are determined by a random value that follows a normal distribution of mean $m$ and standard deviation $sd$. The BS to be discarded from the solution and the ones to be included are randomly selected from the set of available locations.

Once such modifications have been introduced into the solution, a final correction is performed: for each base station location, the fitness of the solution including the transmitter (if it is currently used in the solution) or excluding it (if it is not currently considered) is checked and the best choice is selected for the final solution. This operation is not as intensive as the local search but easily improves the perturbed solution. In order to avoid the introduction of search cycles, the implemented algorithm counts the number of iterations since the last solution update. If $S_{best}$ has not been improved in the last $b$ search iterations, the search strength is increased. Initially, the strength is set to 1 and it is increased by 1 every time the solution has not been improved in the last $b$ iterations. The strength scales the number of elements to be deleted and inserted at each perturbation. It adds more diversification to the search in order to escape from local maxima. If after a number of $i$ strength increases the search continues to be trapped in a cycle, the algorithm is restarted from a newly generated initial solution.

For the intensification of the search, a Hill Climbing local search is introduced into the implementation. The neighborhood of a solution is defined as follows.

1) For each of the available locations that have not been used in the solution, one neighbor that includes an antenna in the corresponding position is created.

2) For each of the available locations where a transmitter has been placed, a first neighbor that excludes such an antenna from the solution is created.

3) For each of the available locations that have been used in the solution, a second neighbor that replaces such a base station with the nearest one is created.

During the local search, a complete neighborhood is generated. From the neighbors obtained, the best one is selected and the process is repeated from the chosen new solution. The process finishes when the local search reaches a local maximum or when the steps are repeated a maximum number of $ms$ times. For the acceptance criterion of the local search and also for the ILS, the solution with greatest fitness is selected. The ILS algorithm has been modified to allow the continuation of the local search when it has not yet arrived at a local maximum and if after its application $s''$ has not improved $S_{best}$. This growth in the search may lead to better local optima.

At the end of the search, the best solution found is returned. The algorithm can introduce some type of memory (history) in order to avoid getting trapped into search cycles.

For the encoding, we have used a binary vector of size $num_{of\ available\ bs\ locations}$ and set “1” if the corresponding BS is used in the solution or “0” if it is not used.

The ILS algorithm was tested in order to tune the set of necessary parameters. After a study the following values were fixed: $m = 3$, $sd = 1$, $ms = 100$, $b = 250$, and $i = 2$.

ILS has been successfully applied to many complex combinatorial problems, especially in timetabling and scheduling [51]–[54] but the number of applications to networks and communication problems is not so extensive [55]–[59].

### D. Population-Based Incremental Learning

Population-based incremental learning (PBIL) is a method that combines genetic algorithms with competitive learning (typical in artificial neural networks) for function optimization [60], [61]. PBIL is an extension of the EGA (equilibrium
genetic algorithm) achieved through reexamination of the performance of the EGA in terms of competitive learning.

PBIL attempts to create a probability vector from which samples can be drawn to produce the next generation’s population. The general process of the algorithm is described in the following pseudocode:

**PBIL Pseudocode**

\[
P \leftarrow \text{InitProbVector} \quad (\text{each position } P_i = 0.5)
\]

while not end condition() do

\[
\text{for } i = 1, \ldots, \text{NS} \text{ do}
\]

\[
\text{sample}_i \leftarrow \text{GenerateSampleAccordingP}()
\]

\[
\text{evaluation}_i \leftarrow \text{Evaluate} \quad (\text{sample}_i)
\]

end for

\[
\text{max} \leftarrow \text{FindSampleWithMaximumEvaluation}()
\]

while LengthProbVectorP do

\[
P_i \leftarrow P_i \ast (1.0 - LR) + \max_i \ast (LR)
\]

end while

while LengthProbVectorP do

if (random \((0, 1) < MUT_P\) them

\[
P_i \leftarrow P_i \ast (1.0 - MUT_A) + \text{random(0.0 or 1.0)} \ast (MUT_A)
\]

end if

end while

end while

return max

As we can see, the necessary parameters for PBIL are the population size (NS, number of samples/individuals to produce per generation), the probability of mutation occurring in each position of the probability vector \((MUT_P)\), the amount for mutation to affect the probability vector \((MUT_A)\), and the learning rate \((LR)\).

Each position in the initial probability vector has the value 0.5 (usual in PBIL). In our case, as we use a binary encoding (there is or is not a BS in this position); this means that both possible values (0 or 1) for each position have, initially, the same probability. After initializing the probability vector \(P\) (each position equal to 0.5), the NS samples (individuals in the population) are generated. Each sample vector must be generated according to probabilities in \(P\). Furthermore, each sample vector is also evaluated using the fitness function. Then, we look for the best sample \(max\). This \(max\) sample is used in order to update the probability vector \(P\), position by position, using the learning rate \(LR\).

Finally, we have to mutate the probability vector \(P\), position by position, using the mutation probability \(MUT_P\) and the mutation amount \(MUT_A\). Preliminary experiments were carried out to find the best set of parameter values for PBIL (in order to solve the RND problem correctly). These values are: \(\text{NS} = 135, MUT_P = 0.02, MUT_A = 0.05,\) and \(LR = 0.10\).

Although PBIL has been used in very diverse optimization problems ([62]–[65]) are some recent examples), surprisingly, it has not been used in many telecommunication studies (only a few cases exist ([66]–[69]).

### E. Clustered Genetic Algorithm

A genetic algorithm (GA) uses a set of genetic operators (selection, crossover, and mutation) to evolve a solution to a problem. The solution is represented as population individuals, and the individuals with higher fitness values have higher probabilities of surviving the selection.

The representation of individuals is one of the most important issues. In the canonical GA, each chromosome is usually represented by a bit string, where each position represents a transmitter \((0 = \text{off}, 1 = \text{on})\). For this problem, we propose using data-mining techniques in order to determine transmitter clusters and only allow one active transmitter in each cluster. This representation makes the search space smaller which is extremely useful. In the reference domain, the binary search space is \(2^{1000} \approx 10^{301}\) and the new search space using 70 clusters is \(10^{70} \approx 10^{84}\). In this proposal, one individual has the same number of genes as clusters. A gene is a list of transmitters in the cluster and a number indicating the active transmitter. Only one BS can be working in each cluster. Inactive BSs are indicated by \(-1\).

**Clustered Genetic Algorithm Pseudocode**

\[
C_i \leftarrow \text{Simple_KMeans}()
\]

\[
P_0 \leftarrow \text{generateInitialSolution}(C_i)
\]

\[
\text{evaluation}(P_0)
\]

while not end condition() do

\[
P' \leftarrow \text{selection}(P_0)
\]

\[
P'' \leftarrow \text{crossover}(P')
\]

\[
P_{n+1} \leftarrow \text{mutation}(P'')
\]

\[
\text{evaluation}(P_{n+1})
\]

end while

2-OPT\((P_n)\)

The first step in the algorithm is to determine the clusters. For this purpose the WEKA implementation of the \(k\)-means algorithm with the employment of Euclidian distance was used [70]. Once the clusters are found, the population is randomly generated and evaluated for the first time. In each successive generation, part of the population is selected to breed a new generation. Both roulette wheel and tournament selections were tested. Roulette wheel selection led us to a premature convergence in some experiments and we decided to use stochastic tournament selection in order to maintain the genetic diversity as much as possible. Hence, tournament selection was used to select which individuals evolve to the next generation. This operator runs a tournament among four individuals chosen at random from the population and selects the one with the best fitness (the winner) for crossover. The selection pressure was adjusted by changing the tournament size in several experiments. In the end, the best results were found from tournaments of four individuals. Uniform crossover selects one gene of each parent alternatively and each child receives 50% of the genetic information of each parent. Mutation occurs according to a user-definable mutation probability swapping the gene value.
Finally, after the stop condition is met, a variant of the 2-OPT (Pairwise Interchange) heuristic is carried out, since it may redefine the solution at a low cost. The variant of the 2-OPT heuristic, also known as the pairwise interchange heuristic [71], has been selected as an LS method. Basically, the 2-OPT technique follows a procedure that searches all the neighbors of the solution looking for better solutions. Briefly explained, the 2-OPT consists of swapping the active antennas given by the CGA (Clustered Genetic Algorithm) with their neighbors and checking whether the final solution improves. These small permutations between antennas can lead to slightly better solutions. To avoid high computational cost, the LS is limited only to the neighbors that are closest.

**F. Clustered Chromosome Appearance Probability Matrix**

The chromosome appearance probability matrix method is a GA modified in order to deal with micro populations. In PBIL algorithms, the recombination operator is replaced by a probability vector for each variable, and sampling this vector requires the study of the selections made by the algorithm until that moment. This concept, applied to interactive evolutionary computation, can be used in order to speed up the evolution according to the user needs. This was the key motivation for developing this new method based on the GA. Basically, it consists of a GA that uses a probability matrix which drives the mutation operator toward the solution, speeding up the convergence during the first generations.

**Clustered Chromosome Appearance Probability Matrix Pseudocode**

\[ C_l \leftarrow \text{Simple}_{-}K\text{Means}() \]
\[ P_0 \leftarrow \text{generateInitialSolution}(C_l) \]
\[ \text{InitializeStatistics}(M_l) \]
\[ \text{evaluation}(P_0) \]
while not end condition() do
\[ P' \leftarrow \text{selection}(P_0) \]
\[ \text{updateProbabilityMatrix}(M_l, a) \]
\[ P'' \leftarrow \text{crossover}(P') ; \]
\[ P''' \leftarrow \text{orientedMutation}(P'', M_l) \]
\[ P_{n+1} \leftarrow \text{cloneRemover}(P''') \]
\[ \text{evaluation}(P_{n+1}) \]
end while
\[ \text{2-OPT}(P_n) \]

The codification is the same as that explained for the GA, and the steps of the proposed algorithm are very similar. The InitializeStatistics() function initializes the probability matrix with all possible combinations of chromosomes. These combinations are calculated by multiplying the maximum sizes of each gene. The probability matrix shows the probability that each possible combination of alleles has of being chosen. The OrientedMutation() function takes a specific chromosome as the base of the mutation process (reference chromosome). This chromosome is selected from all the possible chromosomes following a uniform distribution fixed by the probability array. The higher the value of a chromosome’s likelihood array, the better the probability of being chosen. The cloneRemover() function is responsible for mutating all those individuals which have exactly the same genetic structure as other individuals in the same population.

All pseudocode steps are explained in detail in [72]–[74]. The main modifications for the proposed algorithm are the evaluation, selection, and mutation operators. In addition, a new operator that removes identical individuals and a 2-OPT search (as in CGA) have been included.

**G. Clustered Memetic Algorithm**

The memetic algorithm (MA) is a combination of LS techniques and EAs. It is based on the concept of a *meme* introduced by Dawkins [75]. The key idea of a meme is that an individual can change its genetic code during its life, improving the evolution process. To simulate this concept of a *meme*, it includes an LS in the reproduction operators (crossover and mutation). The MA with cluster representation uses the same codification as the ones described earlier (GAs). The crossover operator is the same as the one used in the GA, but an LS is done in order to find the best possible crossover.

Ideally the LS should calculate all possible crossover combinations and choose the best, but this means too many evaluations per crossover. Therefore the developed LS randomly chooses a predefined percentage of the previously selected individuals (by tournament selection as made in the GA) and then finds the best crossover for those individuals. At this point, when the crossover is done, both parents are marked in order to avoid identical crossovers within the same iteration. The best possible crossover is guaranteed for the selected individuals and the offspring proceeds to the next step.

The mutation operator is the same as the one used in the GA plus an LS. In this case, an LS is done to try to find the best possible mutation. Since we also want to know which permutation of those mutations will improve the solution, we have to test all possible mutated gene combinations. Ideally, all genes should be changed in order to find the best possible mutation. However, as there are too many possible combinations, the operator instead first calculates the number of mutations for each individual and then performs a LS limited to pairs of two genes. This limitation allows the mutation operator to search for the best combination of values for each pair of mutated genes. It reduces the computational costs of evaluating all possible mutation combinations.

**Clustered Memetic Algorithm Pseudocode**

\[ C_l \leftarrow \text{Simple}_{-}K\text{Means}() \]
\[ P_0 \leftarrow \text{generateInitialSolution}(C_l) \]
\[ \text{evaluation}(P_0) ; \]
while not end condition() do
\[ P' \leftarrow \text{Selection}(P_0) \]
\[ P'' \leftarrow \text{CrossoverWithLS}(P', \%PCT) \]
\[ P_{n+1} \leftarrow \text{MutationWithLS}(P'', 2) \]
\[ P_{n+1} \leftarrow \text{BestMutation}(P', 2) ; \]
\[ \text{evaluation}(P_{n+1}) ; \]
end while
\[ \text{2-OPT}(P_n) \]
H. Differential Evolution

Differential evolution (DE) is an algorithm created by Price and Storn [76]. Since 1994, DE has been used for many optimization problems with satisfactory results [77]–[79].

DE is a very simple population-based stochastic function minimizer, which can be categorized into the class of floating-point encoded evolutionary algorithms. It is currently used in a wide range of optimization problems, including multiobjective optimization [80]. Generally, the function to be optimized $f$ is computed by means of optimizing the values of its parameters, where $X$ denotes a vector composed of $n_{\text{param}}$ objective function parameters. As with all population-based evolutionary optimization algorithms, DE handles a population of solutions instead of a single solution for the optimization of a domain-dependant problem. Population $P$ of generation $G$ contains $n_{\text{pop}}$ solution vectors, each one usually known as an individual of the population. Consequently, each vector represents a potential solution for the optimization problem.

At any time, a population $P$ of generation $G'$ contains $n_{\text{pop}}$ individuals, each one containing $n_{\text{param}}$ parameters (usually referred as chromosomes). In order to establish a starting point for seeking an optimum, the population $P^{(0)}$ (initial population) must be initialized. This is usually done by seeding $P^{(0)}$ with random values that are within given boundary constraints.

The population reproduction scheme of DE is different from other evolutionary algorithms. From the first generation onward, the population of the subsequent generation $P^{(G+1)}$ is created in the following way on the basis of the current population $P^{(G)}$. First, a temporary individual (usually referred to as a trial) that can possibly populate the subsequent generation $P^{(G+1)}$ is generated as shown in the following:

$$x_{i, j}^{(G+1)} = \begin{cases} x_{C, i, j}^{(G)} + F \cdot (x_{A, i, j}^{(G)} - x_{B, i, j}^{(G)}) & \text{if } r_{i, j} \leq Cr \\ x_{C, i, j}^{(G)} & \text{if } r_{i, j} > Cr \end{cases}$$

where

- $i = 1, \ldots, n_{\text{pop}}, \quad j = 1, \ldots, n_{\text{param}}$
- $A = 1, \ldots, n_{\text{pop}}, \quad B = 1, \ldots, n_{\text{pop}}, \quad C = 1, \ldots, n_{\text{pop}},$
- $A_i \neq B_i \neq C_i \neq i$
- $Cr \in [0, 1], \quad F \in [0, 2], \quad r \in [0, 1]$.

$x_A$, $x_B$, and $x_C$ are three randomly chosen indexes referring to three individuals in the population. The offspring (known as the trial vector) is produced by subtracting the values of the $x_B$ vector from vector $x_A$. Afterward, the previous values are summed with the values of vector $x_C$.

$F$, $Cr$, and $n_{\text{pop}}$ are DE control parameters that remain constant during the search process. $n_{\text{pop}}$ represents the population size, $F$ is a real valued factor in the range $[0.0, 2.0]$ that controls the amplification of differential variations ($x_A - x_B$ operations), and $Cr$ is a real-valued crossover factor in the range $[0.0, 1.0]$ controlling the probability of choosing the mutated value for $x$ instead of its current value.

The generational scheme of DE also differs from other EAs. Accordingly, each computed trial vector (known as a donor vector) is compared with the target vector. The one with the lower value of the cost function $f_{\text{cos}}(X)$ will remain in the population of the next generation.

Differential Evolution Pseudocode

1. **Initialize** ($P(0)$)

Evaluate($P(0)(X_i)$)

while not end condition() do

   $X_A \leftarrow \text{SelectRandomIndividual}(P(G))$
   $X_B \leftarrow \text{SelectRandomIndividual}(P(G))$
   $X_{\text{TARG}} \leftarrow \text{SelectRandomIndividual}(P(G))$

   Offspring $\leftarrow X_{\text{TARG}}(G) + F \times (x_{A, i}^{(G)} - x_{B, i}^{(G)})$

   if $f_{\text{cos}}(X_{\text{TARG}}) < f_{\text{cos}}(\text{offspring})$
     Evaluate(\text{offspring})

   if offspring better than $X_{\text{TARG}}$
     Replace $X_{\text{TARG}}$ with offspring

End while

Return bestIndividual($X_i, P(G)$)

All design issues took into account the differential-evolution fast convergence, which was proven in previous works [81], and its canonical self-adapting differential mutation operator.

Additionally, we developed a differential mutation operator called nearest point differential mutation which uses a DE differential mutation scheme and enforces the RND hard constraints. Before fitness computation of the trial vector, each gene is checked to see if the location is an available BS location (since differential mutation will create nonlegitimate alleles). If not, it is replaced with the nearest available location (Euclidian distance) not yet in the offspring. Encoding is based on a real-valued vector structure.

From the results produced by previous experiments, we found the optimal values for the control parameters $Cr$ and $F$. $Cr$ was set to 0.3 and $F$ was set to 0.2. In-depth details about RND DE experiments can be found in [5].

I. GRASP

The greedy randomized adaptive search procedure (GRASP) is a recognized metaheuristic that has been used successfully for solving many combinatorial optimization problems [82]–[90].

According to the general literature, GRASP is an iterative process, where each iteration consists of two phases: construction and local search. The construction phase builds an initial solution, while the second phase explores the search space based on the result of the previous phase, hoping to find a better solution. The best solution computed after all GRASP iterations is regarded as the final solution. The next sentences depict the pseudocode for a GRASP procedure.
The GRASP metaheuristic is commonly composed of two main parameters: the number of GRASP iterations Max_Iterations and the initial seed for pseudorandom numeric generation. In this paper the Max_Iterations condition has been replaced by the common 5000000.00 fitness evaluation criterion.

A solution is usually represented as a set of elements. The construction phase starts from an empty set and iteratively adds elements to it until reaching a feasible solution. This is achieved by means of a restricted candidate list (RCL) that integrates all existing elements sorted by function of their problem-specific-dependent myopic greedy evaluator. At each step of the construction phase, the RCL will only be composed of elements that have not been selected to be included in the initial solution. Furthermore, each time an element is added, the cost or fitness of the evolving solution is updated. Usually, the selected candidates are those that induce the smallest increment cost; this represents the greedy component of GRASP. A normal complement is to choose randomly, from the RCL, the next element to be added to the solution, which in turn represents the probabilistic component of GRASP. This allows the building of different feasible solutions at the end of each GRASP iteration.

The solutions returned by the construction phase are not guaranteed to be locally optimal with respect to neighboring concepts. The search phase attempts to improve each initial construction by means of a local search algorithm that iteratively replaces the current solution by a better one. Furthermore, the construction phase plays an important part, since good starting solutions are desirable. There are two basic strategies employed on exploring a solution’s neighborhood.

1) Best-improvement: all neighborhoods are evaluated and the current solution is replaced by the best neighbor.

2) First-improvement: the current solution is replaced when finding the first better neighbor solution.

According to [91], in most cases, when applying both strategies, they achieve the same quality in their final solution, but generally the first improvement strategy takes a lower computational effort. We also observe that it is more common to arrive at premature convergence to a nonglobal local optimum when using best improvement instead of first improvement.

Further details, formal definitions, and GRASP extensions can be found in [82], [85], [91], and [92], which also include extensive analysis of GRASP metaheuristics based on many applications. The internal representation of the GRASP-based approaches uses real-valued encoded elements, where each one represents the coordinates of the BS in the solution.

The RND GRASP-based approach developed is summarily described as follows: Shrinking-RCL (GRASP_SRCL) is a GRASP implementation that uses a tunable greedy local search algorithm. The search procedure uses a canonical RCL greedy mechanism for selecting a new solution iteratively. Each new solution is computed by exchanging an element for another that fits better, if one such is available. An RCL size control parameter is also used to define the greediness of the LS. This implementation uses a continuously shrinking RCL, rendering the search deterministic when sizeof(RCL) = 1, and ending it when sizeof(RCL) = 0. During the iterative search procedure, the RCL shrinks when the loopsize parameter reaches zero, ending its execution when the RCL is empty. The optimal parameters found are RCL (construction phase) = 20, greedy RCLSsize = 5 and Loopsize (LS Phase) = 30.

J. Variable Neighborhood Search

Variable neighborhood search (VNS) is a modern metaheuristic introduced by Mladenovic and Hansen [93] based on systematic changes of the neighborhood search space to solve optimization problems. Its main strategy is based on the employment of more than one neighborhood structure during the search. Its main dynamic focuses on the change of the neighborhood structure in a systematic way as the search progresses. This is one of the most recent metaheuristics developed for solving problems in an easier way. It is acknowledged as being one of the very well-known local search methods [94], [95], getting more attention day by day because of its ease of use and its accomplishments in solving combinatorial optimization problems such as the ones currently being dealt with [96]–[107].

VNS is a simple and effective search procedure that proceeds by a systematic change of neighborhood. A common VNS implementation builds an initial solution \( x \in S \), where \( S \) is the whole set of search space, controlling it through a two-level nested loop in which the core one alters and explores via two main functions named shake and local search. The outer loop works as an energizer, reiterating the inner loop, while the inner loop carries the key search. Local search looks for an enhanced solution within the local neighborhood, while shake diversifies the solution by changing it randomly to another local neighborhood. The inner loop iterates as long as the solutions keep improving, where an integer control parameter \( k \) defines the length of the loop, hence defining the number of shifting neighborhood structures. Once an inner loop is completed, the outer loop re-iterates until the predetermined termination condition is satisfied. Since the set complementarity of neighborhood functions is the key idea behind VNS, the neighborhood structure and the heuristic functions should be carefully chosen to achieve an efficient VNS implementation. Theoretically speaking, intensification is achieved by the local search while the shaking of the neighborhood structure acts as a diversification mechanism, raising its probabilities of avoiding nonglobal optima.

In order to develop an effective VNS algorithm, two kinds of neighborhood functions are required: \( N^I_S(x) \) and \( N^{LS}_S(x) \), each yielding a particular association of neighboring structures,
where $N^k_s(x)$ and $N^{LS}(x)$ denote neighborhood functions for shake and local search functions respectively. It is usually reported [93]–[95] that multiple neighborhood structures may be used for each function (shake and local search), allowing them to achieve different views of the search landscape and allowing the shaking phase to generate new starting solutions that lie near other local optima. For that reason, the indexes $k$ and $l$ are to be used for shake and local search functions, respectively, in order to ease switching from one neighborhood to another.

**VNS Pseudocode**

$x \leftarrow$ Initialize

while not end condition() do

$x' \leftarrow$ GenerateRandomStartingSolution $x' \epsilon N^k_s(x)$.

$x'' \leftarrow$ localSearch($N^{LS}(x)$)

If $x''$ is better than $x$

$x \leftarrow x''$

end while

Return $x$

If the local search uses the greedy strategy, then an iterative procedure tests the entire base, returning the best neighboring solution until a local minimum is obtained. The shake procedure selects a random solution from the global search space.

There are many variants of variable neighborhood search such as *variable neighborhood decomposition search* [108] and *skewed variable neighborhood search* [95]. Given the flexibility of the technique, other variants of this algorithm can be employed [94].

Two VNS derivates were developed to tackle the RND optimization problem. These are described as follows.

**VNS (EVNS)** is a basic canonical VNS implementation with local search. The internal representation of all VNS-based approaches uses real-valued encoded elements, where each one represents the coordinates of the BS in the solution or search space. The LS operator uses the greedy strategy, i.e., it replaces the current element $x'$ with the best one found in the $gn$ neighborhood. If $x'$ is the best element, then no replacement is made. In the case of the RND application, we employed $gn \in \mathbb{R}^+$, where $gn$ is computed by means of an Euclidean distance function, representing the distance between elements that belong to $M$. Since we are working with a coordinate system, the neighborhood structures have been defined on the grid’s coordinate system, employing a Euclidean distance function to define $gn$. The initialization and shaking phases are carried out through a RCL-like construct.

**GRASP VNS (GRASP_VNS)** is a VNS-based implementation that allows dynamic changing of neighborhood range when executing the local search procedure, increasing it when no better neighborhood selections are found and decreasing it while better solutions are continuously found. This is an implementation that uses the GRASP metaheuristic to power the global search (replacing the shaking operator). The LS procedure is the same as in the previous VNS-based description [93]. Experiments were carried out to find the best set of parameter values for the VNS variants. These values are: $gn$ neighborhood size = 45 (150% max. BS radiating distance), shaking RCL size = 5, initial RCL size = 20, and neighborhood variance = 15 (GRASP_VNS only).

**K. Hybrid and Multistart Variants**

**Fixed Neighborhood Tabu (MS_FNS)** is a hybrid LS procedure based on the greedy VNS and tabu techniques, mainly through the prevention of previously visited solutions. The global search space is managed by a multistart mechanism. The additional parameter is the tabu tenure = 25 (fitness evaluations).

As opposed to the previous algorithms, the multistart versions presented next are all derived from previously presented algorithms. Although each of the former search procedures has unique and extendable techniques for avoiding local optima, several of our experiments gave us an insight that some pitfalls could not be avoided using a single modeled technique algorithm (even employing multiple techniques). One of the options considered during development and experimentation was to create multistart versions of some of the promising algorithms that occasionally got trapped in local optima, yielding high deviation values in their final results. The result profiling and the high deviation substantiated that the algorithm could indeed be very effective but also occasionally very deceptive. To avoid these common problems, multistart versions of the following algorithms were implemented:

**Reversed Unleashed Neighborhood Search** (RUFNS) is a triple hybrid searcher, originally based on the GRASP metaheuristic combined with the same VNS neighborhood operations explained in the previous section and a tabu propagation technique, mainly through the prevention of previously visited solutions. The LS operations alternate with the tabu techniques that are employed. This GRASP version delivers heavy-based heuristics while executing the LS phase (as opposed to our light-weighted SRCL_GRASP and GRASP_VNS local search procedures). The additional parameter is the tabu tenure = 25 (fitness evaluations).

**Multistart VNS (MS_VNS)** is a multistart version of EVNS. This version replaces the shaking operator by an MS mechanism.
MS Greedy Entropy Perturbation VNS (MS_GEPVNS) is a hybrid implementation, based on the MS_VNS, employing both a shaking neighborhood operator and a global-scope greedy entropy perturbation mechanism. The greedy entropy perturbation is based on a RCL-like structure and explores the global search space (a control parameter determines the greediness of the entropy). The local search procedure remains unchanged and population elements rely on a simple acquired immune response system (based on the proximity of neighborhood elements). The IN range or neighborhood structure is defined and all \( x_i \in U \cap \text{Neighbors}(x_i, IN) \) are summed. Each element needs to accommodate its immune base level as shown in

\[
x_{\text{immunity}} = IF \left( \frac{\text{Max} \left( |\text{Neighbors}(x', IN)| \right)}{-|\text{Neighbors}(x_i, IN)|} \right),
\]

where \( IF \) represents the immunity amplifying factor control parameter, ranging between \([0,1, \infty]\). High immunity levels decrease convergence speed; hence the importance of adequate balancing of the \( IF \) factor.

Each time a move is made during the LS phase, if the element’s current immunity is higher than zero, it is decreased by 1 and the LS move is not committed.

A multistart mechanism is implemented when stagnation or premature convergence is detected during runtime. This approach delivers a double outer-diversity echelon: 1) by its shaking operator and 2) its multistart uttermost mechanism that relies on an initialization heuristic. While the previous approach emphasizes re-initialization (hence being denoted as GRASP), this one is classified as a multistart variant due to its controllable initialization method. Preliminary experiments were carried out to find the best set of parameter values for MS_GEPVNS. These values are: \( gn \) neighborhood size = 45 (150% max. BS radiating distance), \( IF \) amplification = 1, shaking RCL size = 5, and initial RCL size = 20.

VI. RESULTS

In this section we present and describe the experiments performed with the different optimization algorithms depicted in Section IV, including some advanced distributed runtime environments.

A. Heterogeneous Distributed Environments

Although the approaches studied in this paper aim to reduce the computational effort, the search space is still considerable, and thus we also resorted to high-throughput and grid computing. Wall-clock runtime for all algorithms is bounded between 0.5 and 5h per experiment. Additionally, some techniques combining software and hardware were used in order to accelerate the computations. Some of these are briefly described as follows.

1) BOINC Desktop Grid Computing: Berkeley open infrastructure for network computing (BOINC) [109], [110] is a system for “volunteer computing” and “desktop grid computing.” Volunteer computing uses computers volunteered by the general public to do distributed scientific computing. We used the middleware system BOINC in order to perform thousands different executions of the PBIL algorithm in parallel. In this way, we were able to do a deep survey of which are the best parameters and combinations for solving the RND problem.

2) CONDOR Desktop Computing: The executions of the DE, GRASP, VNS, Hybrids, and MS variant experiments described in this paper were carried out through the Condor high-throughput computing framework [111]. Specifically, the Condor framework permits the harvesting of computing resources that would otherwise be left idle, allowing users with access to the Condor system to submit batches of independent tasks. These tasks are then scheduled by the Condor master over the available computing resources. If a task does not complete in the assigned machine—for instance, the remote machine is taken back for interactive usage or the machine is simply turned off—the execution lease times out after a given time interval and Condor automatically reschedules the task to another machine. All of this is in practice transparent to the application programmer, with application submitters only providing the binary application.

B. Problem Instance and Experimental Planning Disclosure

A real-world-sized problem instance, defined by the geographical layout of the city of Malaga (Spain), was used to test the algorithm performances. This instance, named Malaga1K, represents an urban area of 27.2 km² as shown in Fig. 3. The terrain has been modeled using a 450 × 300 grid, where each point represents a surface of approximately 15 × 15 m. This fine-grained discretization enables us to achieve highly accurate results. A dataset containing 1000 candidate sites for the BSs, and their corresponding coordinates on the grid, is used. The dataset can be found on the website [115]. The cell model for BS coverage, as explained in Section III, is an omnidirectional isotropic model, with a radius of approximately one half kilometer (30 grid points). In this scenario, the maximum coverage that can be attained is 95.522%. There are two major uncovered areas: the sea (at the bottom of Fig. 3), and the mountains (Fig. 3, top).

Our experiments promote thoughtful [116], well-planned, and algorithmic extensive testing, full disclosure of experimental conditions, including the integrity and reproducibility of...
reported results. The most relevant results we currently present are based on the following measures.

1) Effectiveness—Defines the fitness-based quality of the results.

2) Computational Effort—Speed of computation is obviously a key factor. In this paper the computational effort is based on the FEEM definition as shown in Section IV-A. The number of fitness evaluations is used to define when the average maximal fitness of an algorithm is reached.

3) Algorithmic reliability—Defines a degree of confidence for a given algorithm to yield good results, according to its average effectiveness. This is achieved through the standard deviation of the maximal fitness results computed by each algorithm.

Additionally, all results were derived from statistical experimental design techniques aiming at the reduction of variability within the results and promoting a comprehensive report of the results.

For each of the proposed algorithms, 30 independent runs were conducted with a stopping criterion of 5000000 fitness evaluations each. All presented results rely on the statistical values yielded by each of the 30 runs.

C. Normalized Behavior Models

The algorithms can be classified into three quartile groups according to the averaged fitness of the best solution obtained in the 30 runs. The first group (the algorithms producing solutions with the highest fitness) contains MS_GEPVNS, ILS, and CHC. The second group contains MS_FNS, MS_VNS, and HybridGRASP_RUFNS. Finally, the third group (the ones producing the lowest fitness) contains GRASP_EVNS, AGC, MAC, SA, and DE.

Fig. 4 shows the runtime quality distribution of the algorithms in the upper quartile. We observe that the faster the convergence is achieved, the higher the final fitness (which is not a usual occurrence). In this sense, MS_GEPVNS is the algorithm that produces the highest final fitness (164.701) and is also the first to converge (2000000 FEEM), while PBIL gets the lowest fitness (162.651) and is the last to converge (3500000 FEEM), with ILS (164.092 fitness and 1850000 FEEM), and CHC (163.278 fitness and 3 300 000 FEEM) being second and third, respectively.

The median-quality runtime distributions are shown in Fig. 5. Due to their GRASP and multistart-based foundations, these algorithms have a common behavior—all three converge very quickly. HybridGRASP_RUFNS converges faster, after less than 100000 FEEM. MS_FNS and MS_VNS both converge at approximately 500 000 FEEM. HybridGRASP_RUFNS also obtains the highest fitness (162.411), followed by MS_VNS (162.120) and then MS_FNS (161.884).

The lower quartile behavior is shown in Fig. 6. The algorithms in this quartile present different behaviors. We can define three behavior segments: the first segment presents a very fast convergence (GRASP_VNS, DE), the second one presents a very slow convergence (SA), and the third one starts with a fast convergence, then a long stagnation, and finally a sudden fitness rise (AGC, MAC, CAPMC). This sudden rise remains an inexplicable phenomenon. The algorithms AGC, MAC, CAPMC, and GRASP_VNS all produce similar fitness (between 161.352 and 162.134), followed by SA (156.476), while DE and GRASP_SRCL (148.802) have clearly lower results.

D. Effectiveness Comparison

We define the effectiveness of an algorithm as the capacity the algorithm has to achieve good fitness-based results. In this problem instance, the optimal value is unknown, although some approximation can be deduced by the combined comparison of the fitness-based results and their standard deviation.

Fig. 7 depicts the average effectiveness (fitness) achieved per algorithm. This measure is based on the average convergence point $P$ obtained through the $AvgSeries$ runtime distributions where $AvgSeries = \frac{\sum_{i=1}^{30} P_i}{30}$, where Pivot $P = \text{Max}[f(x)]$ and $f(x)$ represents the function to be maximized. There seem to be two comprehensible subsets of algorithms. The first contains high-performance RND algorithms, all producing average fitness values above 161. The second set, formed by DE, GRASP_SRCL, and SA, are classified as outsiders since their fitness fall between 148.196 and 156.478.

In the first subset, the algorithms that get the highest fitness are MS_GEPVNS, ILS, and CHC, respectively.
A commonly used measure is the standard deviation of the fitness of the average convergence point $P'$. Fig. 9 shows, for each algorithm, the $DevSeries$ standard deviation on $P'$ where $DevSeries = StdDev(\sum_{i=1}^{30} Max(f(x_i)))$. The standard deviations are two orders of magnitude smaller than the average fitness. In this sense, the least reliable of the algorithms is DE, with a standard deviation of 1.72% (2.552) of the average fitness. We also observe that all GRASP, multistart, and ILS algorithms have a standard deviation under the 0.5 boundary. There is another interesting detail: the MS_GEPVNS algorithm, besides presenting itself as the most effective algorithm in the set, has a standard deviation of 0. This algorithm thus proves to achieve the maximal result in 100% of conducted experiments. Nothing can be deduced about its optimality, but its standard deviation encourages a good provisional confidence in this matter.

G. Algorithmic Trend Disclosure

Some global patterns or trends have been observed upon analysis. We defined the slope of each of the algorithmic runtime quality distributions based on the slope of the linear regression line for each of the runtime series. These results pointed to a segmented performance ratio between population-based and non-population-based approaches, with the exception of SA (although its results are not outstanding). DE, which is known for being a very fast population-based optimizer [76], clearly defines the boundary between the two segments, also with low quality values, suffering excessively from a phenomena called stagnation [117].

Overall, we can observe that the GRASP approaches are very fast but are also strongly disposed toward local-optima trapping. GRASPs light LSs are unable to explore the search space in an effective way, although better results are observed when the LS procedures incorporate heavier heuristics or additional flow mechanisms (as the HybridGRASP_RUFNS). The combinatorial complexity of real-world RND instances is not satisfactory for GRASP approaches that exclusively emphasize the optimization itself, although these approaches can be used for interactive computer-aided design tools since they deliver reasonable quality in a very short amount of time (in seconds).

By the inspection of Fig. 10, it is possible to observe that LS-based algorithms start with higher fitness values compared to non-LS-based approaches. This is a common behavior in many other problems. The same observation is drawn from the other quartiles.

None of our population-based approaches suffered from premature convergence (except the previously mentioned DE phenomena) since these bio-inspired models intrinsically use their control parameters to avoid such pitfalls, but they also usually turn out to be slower than most of the contestants, although yielding good results. Population-based approaches have the intrinsic burden of having to evaluate all the solutions that compose the population, while other methods like LS-based approaches do not. Since FEEM shows that the evaluation function is extremely computing intensive, this burden is passed on from the implementations and has to be accounted for in real-world applications. On the other hand, these

E. Computational Effort Comparison

In this section we present the FEEM computational effort comparison among the algorithms. The convergence point $P'$ is defined by the FEEM that have elapsed until the best result has been found.

This measure is based on the average convergence point $P$ obtained through the $AvgSeries$ runtime distributions.

Fig. 8 gives an overview of the average convergence point $P'$ obtained through the $AvgSeries$ runtime distributions, including the standard deviation for $P'$ or $Avg[Max(f(x))]$, established through the $DevSeries P$ distributions, where $DevSeries = StdDev(\sum_{i=1}^{30} FEEM(P' ))$.

From an inspection of the figure, the algorithms can be classified into two segments according to their computational effort. The faster segment contains Hybrid_RUFNS, GRASP_SRCL, DE, GRASP_VNS, MS_FNS, MS_GEPVNS, and MS_VNS. On the other hand, the slower segment contains the rest of the algorithms: ILS, CHC, PBIL, SA, CAPMC, AGC, and MAC. Additionally, this segment includes ILS which exhibits a soaring standard deviation to achieve $P'$ according to its convergence speed.

As a final observation, it is possible to state that in a general manner, not a population-based approaches have better computational effort measures, with the exception of DE.

F. Reliability Comparison

The reliability of an algorithm refers to the extent of confidence for a given algorithm to achieve good results in any execution, tightly related to its average effectiveness.
In the future, we will bend our research in two main directions: 1) the inclusion of multiobjective optimization methods in order to enlarge our optimization approach support base and 2) the creation of additional specific instances with landscape simulation features, including path-loss models and bandwidth demand zones.

REFERENCES


[70] I. H. Witten and E. Frank, Data Mining: Practical Machine Learning


Guillermo Molina received the Engineering degree in telecommunications from the University of Malaga, Malaga, Spain, in 2005, where he is currently working toward the Ph.D. degree.

His research topics are optimization problems in networks, notably problems found in radio and wireless sensor networks, the use of novel metaheuristic techniques, and their application to complex real-world scenarios.

Miguel A. Vega-Rodríguez received the Ph.D. degree in computer science from the University of Extremadura, Caceres, Spain.

He is currently an Assistant Professor of computer architecture in the Department of Technologies of Computers and Communications, University of Extremadura. He has authored or co-authored more than 260 publications including journal papers, book chapters, and peer-reviewed conference proceedings. His main research interests are parallel and reconfigurable computing and evolutionary computing.

Juan A. Gómez-Pulido received the Ph.D. degree from the Complutense University of Madrid, Madrid, Spain, in 1993. He is currently an Assistant Professor of computer sciences in the Department of Technologies of Computers and Communications, University of Extremadura, Caceres, Spain. He has authored or co-authored 25 ISI journals, and many book chapters and peer-reviewed conference proceedings. His current research interest is the application of high-performance reconfigurable computing to speed-up evolutionary algorithms in large optimization problems.

Yago Sáez received the degree in computer engineering from Universidad Pontificia de Salamanca, Salamanca, Spain, in 1999, and the Ph.D. degree in computer science from Universidad Politécnica de Madrid, Madrid, Spain, in 2005. He is an assistant professor at the Department of Computer Science, Artificial Intelligence Area, Carlos III University of Madrid, Spain. He takes part of artificial neural network excellence association (REDAF) and the Computational Finance and Economics Technical Committee of the IEEE Computational Intelligence Society. His main research areas encompasses the evolutionary computation, the computational economic and finance applications and the optimization by means of metaheuristics.

Carlos Segura received the Diploma degree in computer science in 2006 from the University of La Laguna, Tenerife, Spain. He is currently a Ph.D. student with the Department of Computer Science, University of La Laguna, where he works as Researcher in the MSTAR project.

His research topics include cutting and communication problems, high-performance computing, and metaheuristics.

Enrique Alba received the Diploma degree in engineering and Ph.D. degree in computer science, both from the University of Malaga, Malaga, Spain, in 1992 and 1999, respectively.

He is currently a Professor of data communications and evolutionary algorithms at graduate and master programs, respectively, with the Department of Languages and Computer Sciences, University of Malaga. He leads a team of seven doctors and eight engineers in the field of complex optimization. He has offered dozens of doctorate courses and multiple seminars and has directed several research projects and contracts for innovation and transference to the industry. He has published many articles in ISI journals and refereed conferences.

Dr. Alba has merited six awards for his professional activities. His H index is 13.

Pedro Isasi is currently a Professor and Head of the Department of Computer Science, Artificial Intelligence Area, Carlos III University of Madrid, Leganés, Spain.

His research is centered in the field of the artificial intelligence, focusing on problems of classification, optimization and machine learning, fundamentally in evolutionary systems, metaheuristics and artificial neural networks.

Dr. Isasi was the Chair of the Computational Finance and Economics Technical Committee of the IEEE Computational Intelligence Society from 2005–2007.

Coromoto León received the M.S. degree in mathematics in 1990 and the Ph.D. degree in computer science in 1996, both from the University of La Laguna, Tenerife, Spain.

She joined to the Department of Statistic, Operational Research and Computation, University of La Laguna, in 1990 as an Associate Professor of compilers. Since 1999, she has been an Assistant Professor for processor languages with the Department of Computer Science, University of La Laguna. She has been the Director of the Spanish national projects TRACER, OPLINK, and MSTAR.

Gara Miranda received the Diploma degree in computer science in 2004 from the University of La Laguna, Tenerife, Spain, and the M.S. degree in computer science in 2006. She is currently a Ph.D. student with the Department of Computer Science, University of La Laguna, Spain, where she works as teaching and research staff in training. She works as a Researcher in the MSTAR project.

Her current research interests include cutting and communication problems, high-performance computing, and metaheuristics.

Juan M. Sánchez-Pérez received the Ph.D. degree in physics from the Complutense University of Madrid, Madrid, Spain, in 1976.

He is currently a Professor of computer architecture in the Department of Technologies of Computers and Communications, University of Extremadura, Caceres, Spain. His research interests are artificial intelligence, logic design, and modern computer architectures.