Swarm and Evolutionary Computation 31 (2016) 24–63

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Regular Paper

A quantum-inspired genetic algorithm for solving the antenna positioning problem

Zakaria Abd El Moiz Dahi*, Chaker Mezioud, Amer Draa

MISC Laboratory, Constantine 2 University – Abdelhamid Mehri, Algeria

ARTICLE INFO

Article history:
Received 16 February 2015
Received in revised form
8 April 2016
Accepted 7 June 2016
Available online 16 June 2016

Keywords:
Quantum-inspired genetic algorithm
Antenna positioning problem
Cellular phone networks
Binary optimisation problems

ABSTRACT

Cellular phone networks are one of today's most popular means of communication. The big popularity and accessibility of the services proposed by these networks have made the mobile industry a field with high standard and competition where service quality is key. Actually, such a quality is strongly bound to the design quality of the networks themselves, where optimisation issues exist at each step. Thus, any process that cannot cope with these problems may alter the design phase and ultimately the service provided. The Antenna Positioning Problem (APP) is one of the most determinant optimisation issues that engineers face during network life cycle. This paper proposes a new variant of the Quantum-Inspired Genetic Algorithm (QIGA) based on a novel quantum gate for solving the APP. In order to assess the scalability, efficiency and robustness of the proposed algorithm, the experiments have been carried out on realistic, synthetic and random benchmarks with different dimensions. Several statistical analysis tests have been carried out as well. State-of-the-art algorithms designed to solve the APP, the Population-Based Incremental Learning (PBIL) and Genetic Algorithm (GA), are taken as a comparison basis. Performance evaluation of the proposed approach proves that it is efficient, robust and scalable; it could outperform both PBIL and GA in many benchmark instances.

1. Introduction

Communication is one of the most active areas of research on both scientific and economic levels. Cellular phone networks are probably the most popular means of communication. The big popularity and accessibility of the services proposed by this type of networks made the telephony industry a field with high competition and standards where service quality is a key factor. Such a quality is strongly bound to the design of the network itself. On the other hand, during planning and managing a cellular phone system, engineers face many challenging optimisation problems [59]. Thus, any process that cannot cope with these problems may alter the design of the network and the quality of service, which is unacceptable for both operators and customers.

These optimisation problems occur during the design phase [53] of any generation of cellular mobile networks; 2G, GPRS, EDGE, 3G, $3G^+$, LTE and 4G. This phase regroups the principal issues of the radio mobile network life cycle which are the Antenna Positioning Problem (APP) and the frequency assignment problem in 2G networks, the cell breathing problem in 3G and $3G^+$ networks, and finally mobility management in 4G and LTE networks [55]. The APP remains the most influential and quality-determining; because a good (bad) antenna positioning may lead to situations of inextricable overlapping, that might make frequency assignment, cell breathing or the mobility management easier (harder) and inevitably facilitates (compromises) the rest of the design phase [53,55].

The APP is an NP-hard binary optimisation problem [57,76]. It consists of covering a given geographical area by using the smallest number of antennas, while achieving the largest cover rate. Like the APP, many other real-world challenging problems in several industrial fields, transportation, packaging and networking are formulated as binary optimisation problems. We cite for instance: the travelling salesman problem [28], the bin packing problem [24], the knapsack problem [73], the permutation flow-shop problem, the job-shop scheduling problem [84], the lot sizing problem [52], the quadratic assignment problem [26] and the feature selection problem [31]. These examples make the need for creating efficient algorithms to solve binary problems such the APP as much challenging and tricky as creating algorithms to solve continuous ones.

On the other hand, bio-inspired computation has provided efficient algorithms to tackle optimisation problems. One of these algorithms is the Quantum-Inspired Evolutionary Algorithm (QIEA). The latter is born from the combination of evolutionary algorithms and

* Corresponding author.
E-mail addresses: zakaria.dahi@univ-constantine2.dz (Z.A.E.M. Dahi), chaker.mezioud@univ-constantine2.dz (C. Mezioud), draa_amer@yahoo.fr (A. Draa).

http://dx.doi.org/10.1016/j.swevo.2016.06.003
2210-6502/

the quantum-inspired mechanics from quantum physics. The aim is to use the quantum principles such as the superposition of states, quantum gates and quantum registers to overcome the high complexity of optimisation problems. The Quantum-Inspired Genetic Algorithm (QIGA) is the first and one of the most promising QIEAs. However, despite its well-established efficiency, the QIGA still suffers from many shortcomings that prevent it from solving complex real-world problems such as the APP.

In this paper, we propose a new QIGA for solving the APP. The experiments are carried out on realistic, synthetic and random instances of the problem with different sizes; to assess the scalability, robustness and efficiency of the proposed approach. Several statistical analysis tests have been conducted to assess the quality of the obtained results. The state-of-the-art algorithms, Population-Based Incremental Learning (PBIL) and the Genetic Algorithm (GA), previously designed to solve the APP, are taken as a comparison basis.

The remainder of the paper is structured as follows. In Section 2, we introduce basic concepts related to the APP and the QIGA. In Section 3, we present the newly proposed approach. Section 4 is dedicated to the presentation of the experimental design and the interpretation and discussion of the obtained experimental results. Finally, we conclude the paper in Section 5.

2. Basic concepts

Basic notions related to the APP, quantum computing and the QIGA are given in this section.

2.1. The antenna positioning problem

In this section, we introduce the modelling of the APP. We opt for the model given by Calegari et al. [60,63].

2.1.1. Problem modelling

The APP aims at finding the smallest subset of base station locations among a set of potential locations in a way to ensure the largest possible coverage for a given geographical area. The APP can be modelled as a graph problem, since it recalls other problems in graph theory such as the minimum dominating set [62,76], the maximum independent set [64] and the unicast set covering problem [19,21].

Let L be the set of all potentially covered areas and M the set of all potential locations of base stations. Suppose G, defined by Formula (1), is a graph in which E is the set of edges in the graph verifying that each transmitter is linked to the area it covers. One seeks the minimum subset of transmitters that covers the maximum surface of a given area. In other words, the objective is to find a subset M' such that $|M'|$ is minimised and $|Neighbours(M', E)|$ defined by Formula (2) is maximised [62]; where *Neighbours* represents the set of the covered areas, and M' represents the set of transmitters used to cover these areas,

$$G = (M \cup L, E) \quad \text{and} \tag{1}$$

$$|Neighbours(M', E)| = \{u \in L | \exists\, v \in M', (u, v) \in E\}. \tag{2}$$

Fig. 1(a) gives the representation of nodes of a graph representing a network (antennas and covered areas), while Fig. 1(b) represents the projection of these nodes on the network. Fig. 1(c) illustrates what the segments of the graph are representing in real networks. Fig. 1(d) goes back to the graph representation by modelling the APP as a graph problem. Finally, Fig. 1(e) is an enhanced representation of the APP using bipartite graphs.

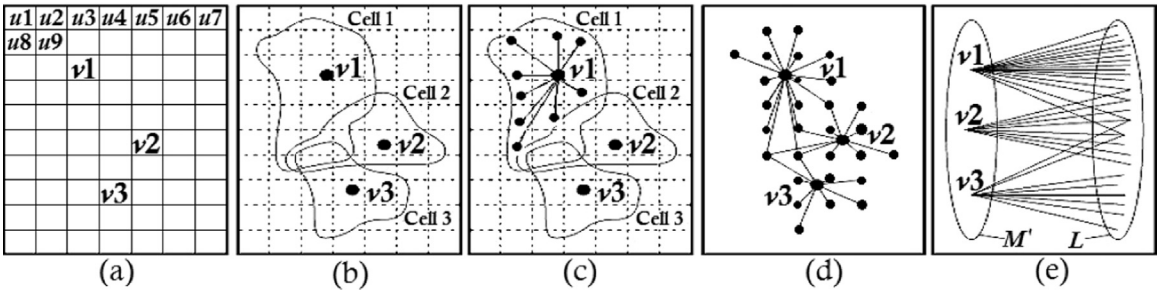


Fig. 1. A graph-based representation of the APP.

A Base Transceiver Station (BTS) is a radio transmitting device with a specific type of coverage (see Fig. 2). In this work, we use three types of coverage introduced in [21]: the squared, omnidirectional and directive coverages. A cell is a part of a geographical area that is covered by a BTS.

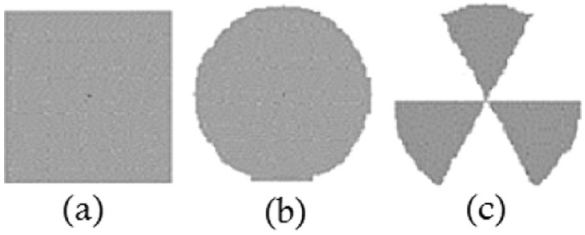


Fig. 2. Antenna coverage models: (a) squared, (b) omnidirectional and (c) directive.

In this work, we address the most practical modelling of the APP with two objectives: maximizing the covered area while minimizing the number of base stations used. The working area is discretised in a rectangular grid with H and V dimensions (see Fig. 3), in which $\{site_1, site_2, site_3, \ldots, site_d\}$ is the set of potentially preselected sites where the antennas can be placed. Each potential site location is identified by its Cartesian coordinates $\{site_1 = (x_1, y_1), site_2 = (x_2, y_2), site_3 = (x_3, y_3), \ldots, site_d = (x_d, y_d)\}$.

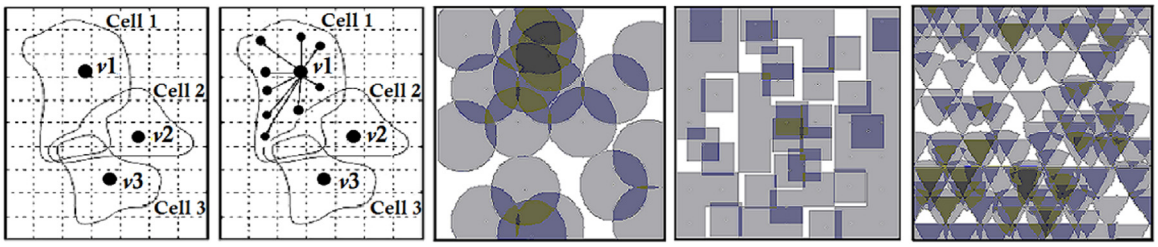


Fig. 3. Representation of the discretised area.

2.1.2. Related work

The first works on the APP were mainly done within the STORMS[1] project and were mostly dedicated to the formulation and the modelling of the problem. The first general formal definition of the APP was introduced in [11], then a more detailed formulation of the problem recalling the minimum dominating set problem was presented later in [63]. In parallel, the first mathematical formulation of the APP was introduced in [60,63], where the authors used a parallel island-based version of the GA to solve it. Another detailed formulation of the APP recalling the unicast set cover problem was given in [62]; the authors of that work used a standard steady state and a parallel island-based GAs. The two previously mentioned formulations were grouped in the work of Kuonen et al. [65], where the authors first used a simple greedy algorithm and then a GA to tackle the APP.

A simple greedy algorithm, a weighted set GA and a parallel island-based GA were used in [64] to solve the APP. Then, a sequential and a parallel steady state GAs were proposed in [19]. A parallel GA was also devised in [20]. A differential evolution algorithm with problem-dependent operators was presented in [79].

A faster differential evolution algorithm with more suitable operators for the APP was proposed in [77], then simulated annealing and the PBIL, the canonical differential evolution and an evolutionary-based algorithm were used in [57] to solve it. In [21], the authors used a simulated annealing, a steady state and a generational GAs and an evolutionary-based algorithm to tackle the APP. Furthermore, a multiobjective evolutionary-based algorithm was proposed in [10] to solve the APP. Several evolutionary algorithms were also presented in [58] to solve it.

Three evolutionary algorithms which are the GA, the memetic algorithm, and the chromosome appearance probability matrix algorithm were used in [86] to solve the APP. The authors of [76,78] used several evolutionary and non-evolutionary algorithms for benchmarking several solvers of the APP. Several multiobjective algorithms such as the non-dominated sorting genetic algorithm II, the strength pareto evolutionary Algorithm 2 and the adaptive and non-adaptive indicator-based evolutionary algorithm were also used in [17] to tackle a multiobjective formulation of the APP. Then, a parallel hyper-heuristic was devised in [18] for the APP. A multiobjective formulation of the latter was proposed in [16]. Several parallel island-based algorithms were also designed in [22] to address the APP; and finally a differential evolution and a binary bat algorithm for solving the APP were recently proposed in [40] and [88], respectively.

2.2. Quantum computing and quantum-inspired algorithms

In this section, firstly we give a timeline appearance review of both quantum computing and QIEAs. Then, some preliminaries of the latter are presented. Finally, we introduce the novel solution representation used within quantum evolutionary algorithms.

2.2.1. Preliminaries and timeline review

Quantum computing is a paradigm born from the rising interest in quantum mechanics in physics. In the early 80s, Richard Feynman observed that some quantum phenomena such as *entanglement* cannot be efficiently simulated on a computer [25]. His observation led to the speculation that computation in general could be done more efficiently if it was using this quantum phenomenon. This speculation proved justified in 1994, when Peter Shor proved that a quantum algorithm for factoring a number takes a polynomial time, while the best-known classical factoring algorithm has a complexity of $O(2^{n^{1/3}\log(n)^{2/3}})$ [69]. On the other hand, Grover's quantum database search algorithm [50] could find an item in an unsorted list of n items in $O(\sqrt{n})$ steps, while classical algorithms require $O(n)$.

Quantum computation is based upon physical principles from quantum mechanics theory such as the *quantum bit*, *quantum register*, *quantum gates* and *superposition of states*. The latter is one of its principal features. Indeed, the *many universes* concept assumes that any object has the ability of being at the same time in several locations according to given probabilities [8]. So, in the late 80s and the early 90s, the idea was to use these quantum principles in order to design methods that can overcome the high dimensionality of combinatorial problems.

Classical computing systems represent a single bit of information deterministically: the value is either 1 or 0. Through the projection of the quantum principles on classical algorithms, the new block of information will have the ability of being 1 or 0 at the same time according to complementary probabilities. This new unit of information called *quantum bit* is a unit of a vector in the complex *Hilbert* space C^2 [5]. On the basis of this, a quantum register of d quantum bits will be able to represent 2^d states at the same time instead of one state when using traditional binary encoding. All the states present within the superposition are processed in parallel when a quantum operation is performed.

Quantum algorithms consist of successively applying a series of quantum operations on a quantum system. Quantum operations are performed using quantum gates and quantum circuits. This characteristic offers an exponential speedup of computation on quantum

[1] STORMS: Software Tools for the Optimisation of Resources in Mobile Systems.

computers compared to their classical counterpart. Yet, there is no powerful quantum machine able to execute the developed quantum algorithms. Therefore, some researchers tried to adapt some properties of quantum computing to classical machines [33].

Since the late 90s, merging quantum computing and evolutionary computation was proven to be promising for tackling complex problems. The first work hybridising these two paradigms goes back to 1996 [8], when Narayanan and Moore proposed the first QIEA. Like any other evolutionary algorithm, a QIEA relies on the representation of individuals, the evaluation function and population dynamics. The particularity of the QIEAs mainly stems from the quantum representation they adopt, which allows representing the superposition of all potential solutions for a given problem. It also stems from the quantum operators they use to evolve the entire population [74].

2.2.2. Solution representation in QIEAs

Several representations can be used to encode solutions in classical evolutionary computation. These representations can be broadly classified as binary, numeric and symbolic. Unlike ordinary evolutionary algorithms, the quantum-inspired ones use a probabilistic representation called the quantum bit *Q-bit*. The latter is the smallest unit of information stored in a two-state quantum computer [83].

A *Q-bit* may be in the state 1, the state 0, or in a linear superposition of both. It is encoded as a pair of probabilities α and β: $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$, where α and β are real numbers that specify the probability amplitudes of the corresponding states. $|\alpha|^2$ specifies the probability of having the state 1 and $|\beta|^2$ specifies the probability of having the state 0 [46]. Normalisation of the state to unity guarantees [45]:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{3}$$

Using the *Bra*/*Ket* notation introduced by Dirac [5,66], the state of each *Q-bit* can be represented as follows:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{4}$$

The *ket* symbol is denoted by $|x\rangle$ and the corresponding *bra* is denoted by $\langle x|$, where $|\Psi\rangle$ denotes more than a vector $\vec{\Psi}$ in some vector space. $|0\rangle$ and $|1\rangle$ represent the classical bit values 0 and 1, respectively [6].

Each quantum individual $\overrightarrow{XQ}$ in the quantum population, *Q-pop*, is generally represented as a *quantum register*. A quantum register is a set of quantum bits, $\overrightarrow{XQ} = \{Q\text{-}bit_1, Q\text{-}bit_2, \ldots, Q\text{-}bit_d\}$, where d is the size of the quantum individual. Finally, a quantum population is a set of quantum individuals, $Q\text{-}pop = \{\overrightarrow{XQ_1}, \overrightarrow{XQ_2}, \ldots, \overrightarrow{XQ_N}\}$, where N is the number of quantum individuals in the quantum population. Fig. 4 shows a typical representation of a quantum population.

$$\begin{array}{c}
\overrightarrow{XQ}_1 \\ \overrightarrow{XQ}_2 \\ \vdots \\ \overrightarrow{XQ}_N
\end{array}
\begin{bmatrix}
\alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \cdots & \alpha_{1,d} \\
\beta_{1,1} & \beta_{1,2} & \beta_{1,3} & \beta_{1,4} & \cdots & \beta_{1,d} \\
\alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \cdots & \alpha_{2,d} \\
\beta_{2,1} & \beta_{2,2} & \beta_{2,3} & \beta_{2,4} & \cdots & \beta_{2,d} \\
\vdots & & & \vdots & & \\
\alpha_{N,1} & \alpha_{N,2} & \alpha_{N,3} & \alpha_{N,4} & \cdots & \alpha_{N,d} \\
\beta_{N,1} & \beta_{N,2} & \beta_{N,3} & \beta_{N,4} & \cdots & \beta_{N,d}
\end{bmatrix}$$

Fig. 4. A typical representation of a quantum population.

2.3. The quantum-inspired genetic algorithm

In this section, we introduce the basic steps of the canonical QIGA [46], as well as its related work.

2.3.1. Algorithm description

The QIGA is based on the basic skeleton of the classical GA. Thus, it shares with its classical counterpart some common phases like the initialisation of solutions, selection, variation operators (crossover and mutation), evaluation and finally replacement. Besides, the QIGA has other specific features such as *quantum interference* and *measurement*.

A typical iteration of the QIGA consists of the cyclic application of selection, variation operators (quantum crossover, quantum mutation and quantum interference), quantum measurement, evaluation and replacement. The pseudo-code of Algorithm 1 summarises the QIGA steps.

Algorithm 1. The quantum-inspired genetic algorithm.

```
Initialisation.
while termination criterion is not reached do
    Selection.
    Variation operators: quantum crossover, quantum mutation and quantum interference.
    Quantum measurement.
    Evaluation.
    Replacement.
end while
```

In the following paragraphs, a more detailed description of the QIGA operators and features is given. It is also worth to mention that when referring to the canonical GA in this section, we point to its binary-coded version.

- *Initialisation*: Like in the conventional GA, initialisation in the QIGA is a process where the population is set, evaluated and the best individual is extracted. A given number N of quantum individuals are created in which complementary probabilities α and β are assigned to each *Q-bit*. Then, the entire population undergoes a *measurement* operation (see measurement section) in order to obtain the corresponding binary state $\overrightarrow{X_B}$ of each quantum individual $\overrightarrow{X_Q}$. Finally, all measured individuals $\overrightarrow{X_B}$ are evaluated (see evaluation section) in order to extract the best binary individual $\overrightarrow{X_{B_*}}$.
  As in the classical GA, several types of initialisation also exist in the QIGA. We can cite, for instance, random and heuristic-based initialisation [81].
- *Selection*: In this step, a given number M of quantum parents are selected to create the *mating pool*. The latter will breed using a series of variation operators such as quantum crossover, quantum mutation and quantum interference in order to create $(N - M)$ quantum offspring. Theoretically speaking, all selection strategies of the classical GA are applicable on its quantum counterpart. Several selection strategies exist in the literature. We can cite for instance the roulette wheel and binary tournament. They are based on several schemes such as ranking and tournament, but they all aim at selecting the best individuals for the next breeding process.
- *Quantum crossover*: Once the mating pool created, the quantum parents undergo a series of variation and breeding operations, where crossover is one of them. Like classical crossover, the number of quantum crossover operations is ruled by a given probability P_c.
  Theoretically speaking, all existing types of crossover of the canonical GA are applicable to its quantum counterpart. We can cite for instance the single-point, uniform and arithmetic crossovers. Taking the two-point crossover as an example, its corresponding quantum version consists of exchanging tails and heads of two quantum parents at two randomly chosen switching points to produce two new quantum offspring [44]. Fig. 5 illustrates how a two-point quantum crossover works.

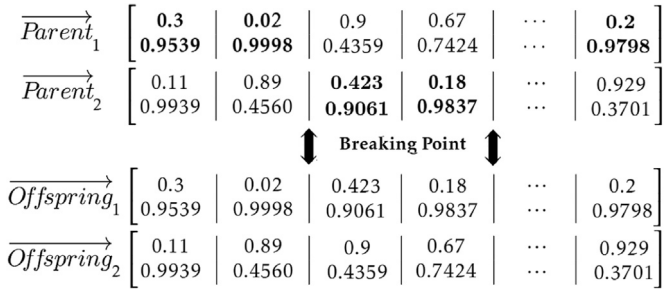


Fig. 5. Two-point quantum crossover.

- *Quantum mutation*: The produced quantum offspring obtained from quantum crossover go through a quantum mutation operation. Similar to crossover, all mutation types of the canonical GA are applicable to the quantum-inspired one. Thus, like basic mutation used in the conventional GA, quantum mutation is performed according to a given probability P_m.
  Several types of mutation exist. We can cite for instance the Gaussian mutation, uniform mutation and Cauchy mutation. For example, a quantum bit-flip mutation consists of flipping in a *Q-bit*, the probabilities of getting a 1 and 0 [44]. Fig. 6 illustrates the working mechanism of the quantum bit-flip mutation.

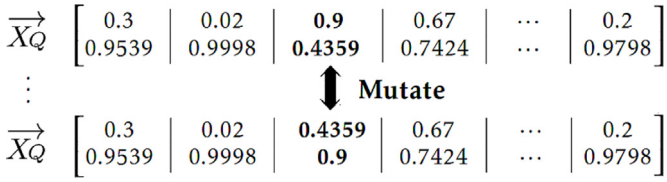


Fig. 6. Quantum bit-flip mutation.

- *Quantum interference*: On the basis of quantum mechanics, a perturbation mechanism called *Q-gate* is used. It is applied to each produced quantum offspring and parent. It guides the latter toward the state $\overrightarrow{X_{B_*}}$ produced by the best quantum individual $\overrightarrow{X_{Q_*}}$. In other words, it amplifies the amplitudes of having the best solution and consequently decreases the amplitudes of having other solutions. So, the quantum interference can also be seen as an intensification phase that encourages better *exploitation* of the best solutions.
  Several *Q-gates* exist in the literature, some of them act like classical logic gates, moving probability from one computational basis state to another, while others produce the non-classical effects of superposition and interference that are responsible for the unique efficiency of quantum computation [48]. The *rotation gate* acting on a single *Q-bit* is the basic *Q-gate* in quantum-inspired algorithms [49]. It is described as given in the following formula:

$$R(\Delta\theta_j) = \begin{bmatrix} \cos(\Delta\theta_j) & -\sin(\Delta\theta_j) \\ \sin(\Delta\theta_j) & \cos(\Delta\theta_j) \end{bmatrix} \tag{5}$$

The variable $\Delta\theta_j$ is a rotation angle that makes the jth *Q-bit* converge toward the state 0 or 1 depending on its objective sign, where $j = 1, 2, \ldots, d$ and d is the size of the quantum individual. After the rotation gate $R(\Delta\theta_j)$ is applied on a *Q-bit* (α_j, β_j), the updated jth *Q-bit*

(α'_j, β'_j) is obtained by applying the following formula:

$$(\alpha'_j, \beta'_j) = R(\Delta\theta_j)\cdot(\alpha_j, \beta_j) \tag{6}$$

It is worth to mention that generally $\Delta\theta$ is designed in compliance with the application problem and each *Q-bit* possibly matches with different angles. Fig. 7 illustrates the working mechanism of the classical rotation gate.

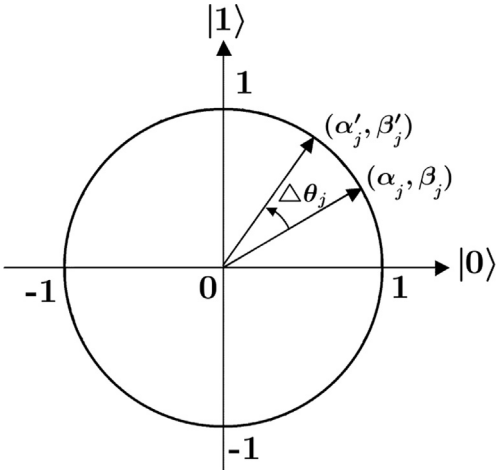


Fig. 7. Quantum interference: rotation gate.

Table 1 gives the value of the rotation angle in function of (α,β) and also the best solution bit value.

Table 1
Lookup table of the rotation angle.

α	β	Reference bit value	Rotation angle
>0	>0	1	$+\Delta\theta$
>0	>0	0	$-\Delta\theta$
>0	<0	1	$-\Delta\theta$
>0	<0	0	$+\Delta\theta$
<0	>0	1	$-\Delta\theta$
<0	>0	0	$+\Delta\theta$
<0	<0	1	$+\Delta\theta$
<0	<0	0	$-\Delta\theta$

- *Measurement*: In binary optimisation problems the solution is encoded as a binary string. On the other hand, the quantum individual is encoded as a probability vector with real values. Thus, a transformation or a mapping is needed to map quantum solutions into binary ones. Measurement is a mechanism that allows generating from each quantum individual $\overrightarrow{X_Q}$ (parents and offspring) a binary solution $\overrightarrow{X_B}$ (i.e. state) among all possible solutions that it contains. But, contrary to pure quantum systems, this measurement does not destroy the superposition of states. So, even knowing that we operate on a traditional machine, it has the advantage of preserving the *superposition of states* for next iterations [6].
  Measurement can also be seen as a *diversification* operator. Indeed, two successive measurements do not necessarily give the same solution, which increases the diversity of the quantum population. For each *Q-bit* $\binom{\alpha}{\beta}$ of the quantum individual $\overrightarrow{X_Q}$, a random number *rand* is generated from the standard uniform distribution $\mathcal{U}(0,1)$. Then, condition (7) is applied. Fig. 8 is an illustration of how quantum measurement performs:

$$X_B = \begin{cases} 1, & \text{if } rand \le |\alpha|^2 \\ 0, & \text{else} \end{cases} \tag{7}$$

$$\overrightarrow{X_Q} \quad \left[\begin{array}{c|c|c|c|c|c} 0.3 & 0.02 & 0.9 & 0.67 & \cdots & 0.2 \\ 0.9539 & 0.9998 & 0.4359 & 0.7424 & \cdots & 0.9798 \end{array}\right]$$

$$\overrightarrow{rand} \quad \left[0.1 \mid 0.29 \mid 0.9 \mid 0.91 \mid \cdots \mid 0.001\right]$$

$$\overrightarrow{X_B} \quad \left[0 \mid 0 \mid 0 \mid 0 \mid \cdots \mid 1\right]$$

Fig. 8. Quantum measurement.

- *Evaluation*: After applying quantum variation operators and quantum measurement, an evaluation phase is performed in order to evaluate the binary individuals obtained from measurement of the newly produced quantum individuals. It is done using the fitness function of the problem being addressed.
- *Replacement*: The replacement step is performed once the evaluation phase is done. It is a process by which the composition of the population for the next iteration is decided. The replacement strategies of the classical GA are all applicable to its quantum-inspired counterpart. We can cite, for instance, $(1+1)$, (μ,λ) and $(\mu+\lambda)$ strategies.

2.3.2. Related work

It is worth to mention that most of the works in the literature use the name *quantum-inspired genetic algorithm*, but in reality few of them are truly genetic. In fact, they use either only mutation, or crossover or no genetic operators at all. So, in such a context, they are in fact more evolutionary algorithms than QIGAs.

Back in 1996, the first QIGA was proposed by Narayanan and Moore in [8]. Since then, it was used to address a large set of problems. A sequential and parallel version of it was devised for solving the knapsack problem in [44,47,80], respectively. Then, it was used in [35] to tackle the calculation of the partition function. Later, the authors in [36,37] proposed a QIGA for the blind source separation problem in signal processing. Then, the infinite impulse response digital filter design was solved in [29] using a parallel version of the QIGA. After, a new version of it based on chaos-updated-rotated gates was introduced in [15]. A QIGA was also used in [82] to solve the control of design process. Then, a new version of it was presented in [33] to tackle the travelling salesman problem. Finally, a QIGA was devised in [32] for image registration.

Recently, the QIGA has been successfully used to solve hard and complex optimisation problems in several fields such as medicine, bioinformatics, image processing, industrial problems, networking, communication, transportation, robotics, information systems and data processing. The authors in [56] used it to tackle the data partitioning problem. Then, the authors in [27,39] applied it to solve networking and communication problems. Later, several academic-combinatorial problems (Knapsack, Max Sat) were also tackled in [70] using the QIGA. The latter was then successfully applied in [85] to solve electromagnetic and wave propagation problems. The authors in [72] used at their turn the QIGA for image processing. It was also used in [14] to address agricultural problems. A QIGA was proposed in [87] to solve a large set of benchmark functions. Finally, a QIGA was also used to tackle industrial problems in [43].

3. The proposed approach

In this section, we introduce the QIGA proposed for solving the APP. Firstly, we introduce the motivation behind our proposed approach. Secondly, we present the newly proposed quantum gate. Then, we present the canonical solution representation and problem formulation of the APP. Finally, the search process of the proposed approach, its main features and how it is applied to the APP are explained.

3.1. Motivation

The GA is a powerful search technique that has been successfully used to solve complex problems in numerous domains of science and engineering [29]. However, it has shown some limits such as large calculation storage, non-convergence to a global optimum and premature convergence [30].

To overcome these limits, a new version of the GA called QIGA, based on the quantum principles was proposed. Many studies showed that the QIGA has superior capabilities compared to its classical counterpart. It has been proven that the QIGA can offer better convergence, diversity and global search capability than conventional GA [36,44,47]. In addition, the authors in [7] showed that some QIGAs have lower complexity, $O(1)$, than their classical counterpart, $O(N\log N)$, where N is the size of population.

The particularity of the QIGA stems mainly from the quantum representation it adopts, which allows representing the superposition of all potential solutions for a given problem. It also stems from the quantum operators it uses to evolve the entire population.

A quantum individual, thanks to its probabilistic representation, can represent 2^d linear states using d *Q-bits*, while in its classical counterpart, each individual represents only one state [13,34]. Thus, a quantum population *Q-pop* can be exponentially larger than a classical population of the same size. These factors allow the QIGA to overcome high complexity of combinatorial problems search space and shorten the calculation time needed to get optimal solutions [12,38].

However, the QIGA still has some shortcomings. Some works showed that it may diverge or get premature convergence. In addition, when addressing multimodal functions, the population diversity tends to gradually disappear and may make the algorithm stagnate in local optima [30,41,87].

Previous works showed that the canonical quantum gate used in the QIGA, the rotation gate, induces the convergence of each *Q-bit* to either 0 or 1 and the latter cannot escape these states. In other words, if the value of $|\beta|^2$ is 0 (or 1), the state of the *Q-bit* will have high probability to remain 0 (or 1) [46,54,75]. On the basis of these observations, we can see that the value of θ greatly influences its performances. In fact, a big value of θ would lead the QIGA to premature convergence, while a small value generally results in slow convergence [87].

These observations can also be noticed in another popular quantum gate which is the H_ϵ gate. From the results of the one-max problem and those of the Schwefel function conducted in [46], it should be noted that if ϵ is too big, the mechanism for exploitation of the algorithm may not work. This is due to the fact that the convergence tendency of a *Q-bit* disappears [46] and induces slow convergence. Taking the classical rotation and H_ϵ gates as examples, we can see that the quantum gate is key in QIGA [41].

3.2. A novel quantum gate: the learning gate

On the basis of the analysis of the canonical QIGA, one can conclude that the basic behaviour of the QIGA is not very suitable for tackling the APP, since the basic operators such as the quantum gate could make the QIGA prematurely converge when facing multiple basins of local optima. This led us to propose a novel interference mechanism more suitable to overcome this kind of issues.

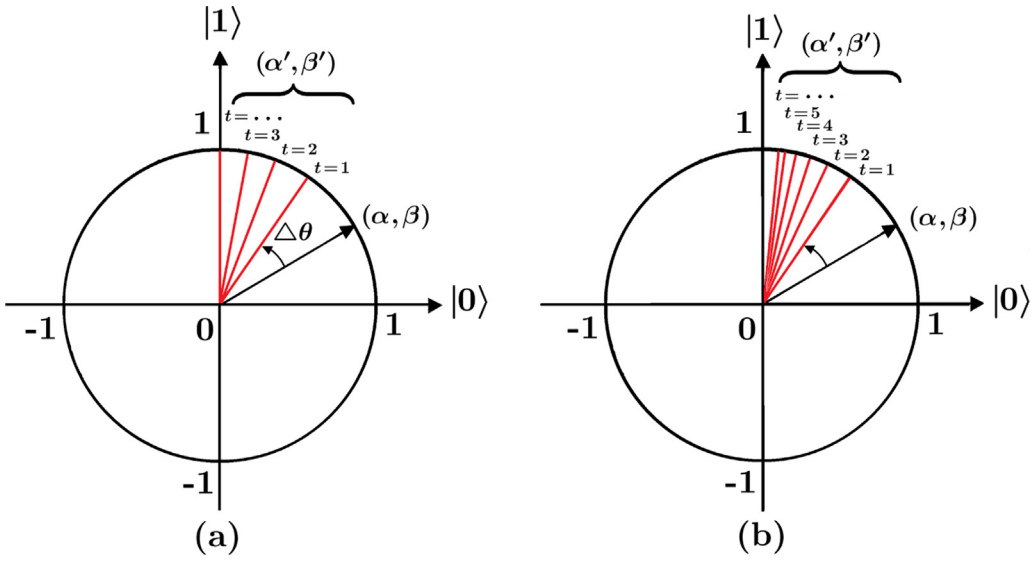


Fig. 9. (a) Rotation gate vs (b) learning gate.

Our new interference formula is different from Formula (6) describing the classical rotation gate, or the one given in [46] describing the H_ϵ gate, in that it *prevents the loss of state superposition*. It can be well-seen from Fig. 9(a) that through time the conventional interference operator makes the superposition of states vanish, the *Q-bit* becomes either a 1 or a 0. On the other hand, as illustrated in Fig. 9(b), the interference operator proposed in this paper allows keeping the state superposition.

Mathematically speaking, the new quantum interference is designed in a way to prevent the stagnation of the algorithm. Let us assume that in the first iteration of the algorithm, a given *Q-bit* $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ in a quantum individual $\overrightarrow{X_Q}$ had 50% chance of being in the state 1 and 50% to be in the state 0 ($\alpha = \beta = \frac{1}{\sqrt{2}}$). Through iterations, this probability will promote the appearance of a state rather than the other, which will lead to stagnation.

In order to cope with this issue, the idea is to subdivide the distance between the current amplitude of being 1 or 0 and the probability of being only 1 or 0. This will allow the quantum individual to keep a percentage of being 0 or 1 even if the actual amplitude of being in the opposite state is much greater, and thus prevents the quantum population from converging to the same state. Fig. 9 illustrates the difference between the classical and the novel interference gates.

Thus, the new variation mechanism proposed in this paper is described by Formula (8), where $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, d$. N is the size of the quantum population *Q-pop* and d is the size of the quantum individual $\overrightarrow{X_Q}$ (i.e. problem dimension). $\overrightarrow{X_{B_*}}$ is the best binary individual obtained so far. *LR* is an inertia weight that prevents the QIGA from quick convergence:

$$\alpha'_{ij} = \frac{LR_*(\alpha_{ij}) + X_{B_*j}}{LR + 1} \tag{8}$$

3.3. Problem formulation

In this section, we present the solution representation and problem formulation proposed by Calegari et al. [60,63] within their modelling of the APP. The latter is formulated as a binary optimisation problem and proven to be NP-hard [76,78].

3.3.1. Representation of solutions

A potential solution of the APP can be represented as a vector of integers described as follows. Each vector $\overrightarrow{X} = \{x_1, \ldots, x_d\}$ represents a potential configuration of the mobile network. The number of elements, d, of each vector represents the number of available candidate sites.

The rank of each element in the solution represents the rank of the corresponding base station in the network (i.e. $j=1$ in the solution vector represents the first BTS in the network, $j=2$ for the second BTS, …, $j=d$ for the last BTS). Each element is strictly binary-valued: $x_j \in \overrightarrow{X} / x_j \in \{0, 1\}$. If $x_j = 1$, the jth base station is selected, otherwise it is discarded. Fig. 10 is an illustration of this representation.

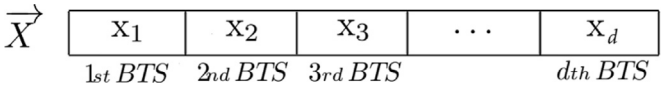


Fig. 10. A typical solution representation.

The vector in Fig. 11 is an example of a potential instance of the APP solution. The solution represents a configuration of a network with eight available locations, where the 1st, 4th and the 8th BTSs are selected, while the remaining BTSs are discarded.

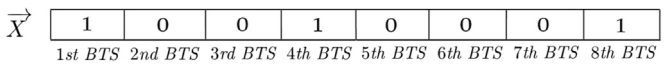


Fig. 11. Example of solution representation.

3.3.2. Objective function

The modelling of the APP proposed by Calegari et al. [60,63] was given within the European STORMS project[2] [60–65,67]. The objective of that project was to produce a set of flexible and partially automated tools to aid in the design and optimisation of UMTS[3] networks and beyond.

Mathematically speaking, several assumptions and criteria have been considered when formulating the APP. Firstly, as stated by the authors of [76,78], this model was designed on the technology-independence principle in order to allow its application on different generations: 2G, GPRS, EDGE, 3G, $3G^{+}$, LTE and 4G. This was achieved by neglecting any of the additional technological constraints that would be thrown into the problem, as for instance, the base transceiver station properties (e.g. azimuth, tilt, path loss models, bandwidth zone prediction, etc.).

In addition, as stated by Calegari in [65], in order to be able to tackle this hard problem, some assumptions had to be made such as *uniform traffic* (i.e. focusing on the coverage problem by assuming that the traffic is uniform all over the area), the *constant cost* (i.e. the total cost of the network is mainly determined by the number of used BTSs). In addition, the model focuses on *rural areas*.

Finally, this field of research actually focuses on the principle of minimisation of resources rather than on achieving the total coverage of an area, since in most real-world problem cases the latter scenario is uncommon [76,78]. Thus, as stated in [61,63,64,67], the APP was brought to its most practical form which is finding out the best possible sites for the base transceiver stations, while ensuring that all or at least a given percentage of the surface of the area is covered and that the global cost of the resulting network is kept at a minimum. Assuming that a set of potential sites is available, the goal is to select the best subset of sites capable of satisfying coverage requirements.

First works conducted on the APP using sequential and parallel GAs [20,60–64,68] gave a first preview of its nature. Then, more recent investigations [10,57,58,76–78,86] using other metaheuristics gave a more accurate idea about the type of the APP. It has been proved in those studies that it is a *non-convex multi-modal* optimisation problem [51].

The APP objective function introduced by Calegari et al. [60,63] was formulated to mathematically transcribe the assumptions and constraints considered during the modelling phase. It is defined by Formula (9):

$$\max_{\vec{X}=\{x_1,\ldots,x_d\}} \frac{\left(\frac{\sum_{i=1}^{V}\sum_{j=1}^{H} S_{ij}}{H*V}*100\right)^{\sigma}}{\sum_{k=1}^{d} x_k} \tag{9}$$

$$\text{subject to} \quad S_{ij} \in \{0, 1\}, \quad i = 1, \ldots, V, \quad j = 1, \ldots, H \tag{10}$$

$$x_k \in \{0, 1\}, \quad k = 1, \ldots, d. \tag{11}$$

The APP objective function defined by Formula (9) is used to assess the quality of a given solution vector $\vec{X} = \{x_1, \ldots, x_d\}$, where H and V are, respectively, the horizontal and vertical dimensions of the discretised grid representing the working area. The variable S_{ij} represents the state (covered and uncovered) of the (i, j)th cell from the discretised grid, where $i=1, \ldots, V$ and $j=1, \ldots, H$. The variable x_k corresponds to the state (placed and not placed) of the kth antenna in the network, where $k=1, \ldots, d$ and d is the number of available sites in the network. σ is a parameter that reflects the importance given to coverage.

The variable S can take one of two values; 1 if the cell is covered by an antenna and 0 otherwise. To determine if a cell is covered or not, the coverage range of the antenna (i.e. radius) (see Table 3) is taken into account. When an antenna is selected to be deployed, every cell in the perimeter of its radius, as well as the cell where the antenna is placed, is considered as covered. It is worth to mention also that a cell can be covered by one or many antennas. In this case, the S value of that cell is equal to 1, no matter the number of antennas that cover it. The variable x can take one of two values; 1 if the antenna is placed and 0 otherwise. It is to be noted that values of H, V and the radius of an antenna are expressed in terms of cells.

As far as the parameter σ is concerned, some researchers such as Segura [17] stated that a decision maker must select its value. It is tuned with respect to the importance given to coverage and the relation with the number of deployed BTSs. However, since there is usually no information about the quality of solutions that can be achieved, such a task is very difficult. So, obtaining the desired solutions is hard and probably several values of σ must be tested. For this reason, other works, such as [65], stated that σ should be greater than 1 in order to favour coverage. The same authors stated later in [68] that σ can be tuned to favour service ratio with respect to the number of used BTSs.

The authors of that work showed that for a value of σ between 0.5 and 1 only one BTS is chosen (that with the largest cell), for σ between 1 and 1.5 coverage is not acceptable (<55%), for σ between 1.5 and 4 coverage grows quickly, while the number of BTSs grows slowly. Finally, for σ between 4 and 10 coverage remains almost constant, while the number of BTSs keeps growing.

A good value for σ should be chosen from the interval [1.5, 4]. Further works, such as [60,62,63], stated that the parameter σ can be tuned to favour cover rate with respect to the number of transmitters. They also stated that so far, telecommunication companies have considered that the results obtained with $\sigma=2$ are of best quality. With respect to these findings, all the works achieved later used the value $\sigma=2$ [10,16,19–22]. Recently, many works treating the APP used the value $\sigma=2$ as a default value [40,57,58,76–79,86].

Since the use of the value $\sigma=2$ has already been discussed and demonstrated in the literature, we use it in this work for scientific relevance and for comparison purposes with state-of-the-art algorithms.

3.4. The search process

A quantum population of N quantum individuals is initialised in the beginning of the search process. Then, each quantum individual is evolved toward a fitter state until a given stop criterion is reached. In each iteration of the algorithm, a cyclic application of a series of variation and evaluation steps is performed. Fig. 12 illustrates the general framework of the proposed approach.

[2] STORMS: Software Tools for the Optimisation of Resources in Mobile Systems.
[3] UMTS: Universal Mobile Telecommunication System or the so-called Pre-3G.

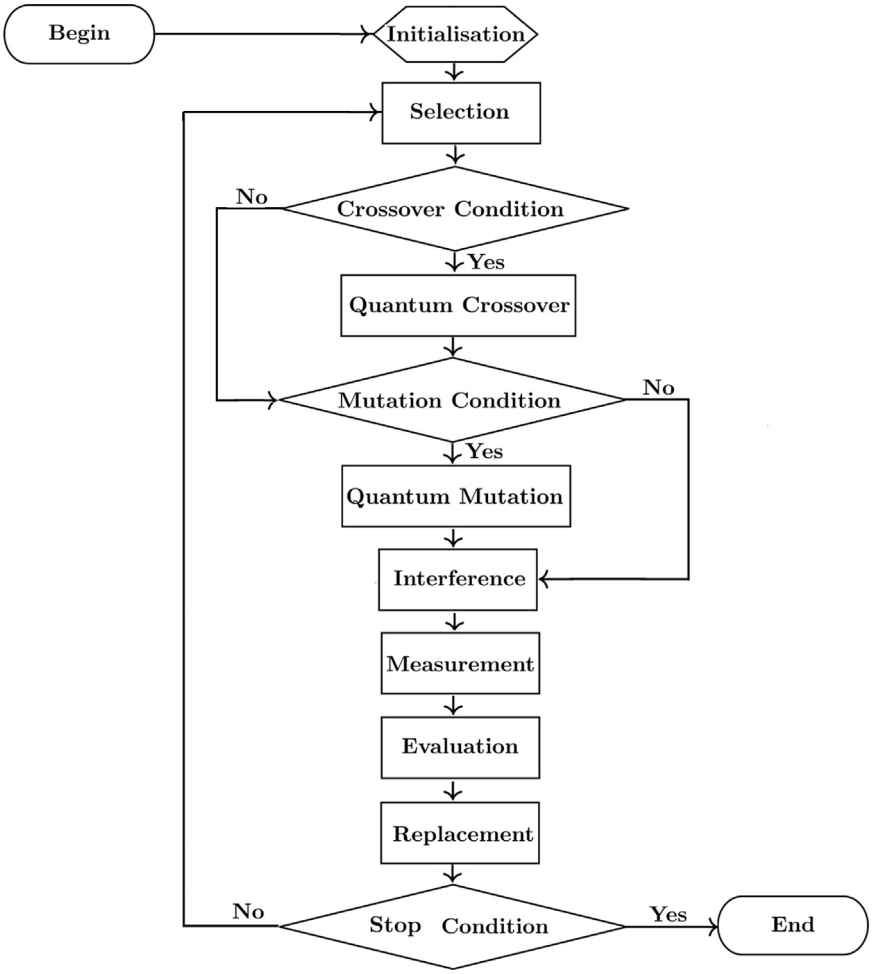


Fig. 12. Flowchart of the proposed approach.

In the following, more insight of the principal steps of the proposed approach is given.

3.4.1. Initialisation

The proposed approach takes as an input a population of quantum individuals *Q-pop*. Each quantum individual $\overrightarrow{XQ}$ of the population represents a superposition of states. Each state is a potential configuration of the network (i.e. solution). The size of the individual represents the number of available locations where BTSs can be placed. Each *Q-bit* in $\overrightarrow{XQ}$ represents a BTS in the network. It is encoded with a pair of complementary probabilities α and β that verifies Formula (3), where $|\alpha|^2$ specifies the probability of having a BTS selected and $|\beta|^2$ specifies the probability of having a BTS discarded.

Fig. 13 illustrates a typical quantum population for the case of the APP, where $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, d$. N is the number of quantum individuals in the quantum population and d is the size of the quantum individual.

The initialisation performed within the proposed approach is the same as the one explained in Section 2.3.1. The difference is that unlike the random initialisation, in our work we set probabilities α and β of each *Q-bit* in the quantum population to the same value: $\frac{1}{\sqrt{2}}$. So, each *Q-bit* will initially have equal probabilities to be in the states 1 and 0 (i.e. every antenna in the network will have initially equal probabilities to be selected or not).

$$
\begin{array}{l}
\overrightarrow{XQ}_1 \left[\begin{array}{c|c|c|c|c|c} \alpha_{1,1} \\ \beta_{1,1} \end{array}\right. \begin{array}{c|c|c|c|c} \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \cdots & \alpha_{1,d} \\ \beta_{1,2} & \beta_{1,3} & \beta_{1,4} & \cdots & \beta_{1,d} \end{array} \left.\right] \\
\quad\quad 1st\ BTS \quad 2nd\ BTS \quad 3rd\ BTS \quad 4th\ BTS \quad\quad dth\ BTS \\
\overrightarrow{XQ}_2 \left[\begin{array}{c|c|c|c|c|c} \alpha_{2,1} \\ \beta_{2,1} \end{array}\right. \begin{array}{c|c|c|c|c} \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \cdots & \alpha_{2,d} \\ \beta_{2,2} & \beta_{2,3} & \beta_{2,4} & \cdots & \beta_{2,d} \end{array} \left.\right] \\
\quad\quad 1st\ BTS \quad 2nd\ BTS \quad 3rd\ BTS \quad 4th\ BTS \quad\quad dth\ BTS \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\overrightarrow{XQ}_N \left[\begin{array}{c|c|c|c|c|c} \alpha_{N,1} \\ \beta_{N,1} \end{array}\right. \begin{array}{c|c|c|c|c} \alpha_{N,2} & \alpha_{N,3} & \alpha_{N,4} & \cdots & \alpha_{N,d} \\ \beta_{N,2} & \beta_{N,3} & \beta_{N,4} & \cdots & \beta_{N,d} \end{array} \left.\right] \\
\quad\quad 1st\ BTS \quad 2nd\ BTS \quad 3rd\ BTS \quad 4th\ BTS \quad\quad dth\ BTS
\end{array}
$$

Fig. 13. Representation of a quantum population for the APP.

3.4.2. Selection

Once the quantum population is initialised, we select half of it to create the mating pool. The latter will contain the quantum parents responsible for creating the new quantum offspring. The classical roulette wheel selection strategy used in the canonical GA is also used within our approach [81].

3.4.3. Quantum crossover

Having the mating pool created, $\frac{N}{4}$ pairs of quantum parents are created randomly to produce $\frac{N}{2}$ new quantum offspring. Each couple of parents is subject to a quantum crossover. The latter is ruled by a probability P_c. A two-point quantum crossover (see Section 2.3.1) is used within the proposed approach [81].

3.4.4. Quantum mutation

The offspring resulting from the quantum crossover operation will be subject to a quantum mutation. Mutation is ruled by a probability P_m. A quantum bit-flip mutation (see Section 2.3.1) is used within the proposed approach [81].

3.4.5. Quantum interference

The interference step is performed on the $\frac{N}{2}$ selected parents and the $\frac{N}{2}$ produced quantum offspring. It is done as described in Section 2.3.1 using Formula (8) of the novel learning gate presented in Section 3.2.

3.4.6. Measurement and evaluation

After the quantum interference step, the newly produced N quantum individuals (i.e. union of parents and offspring) are measured in order to produce binary individuals $\overrightarrow{X_B}$. The measurement done within the proposed approach is the same as described in Section 2.3.1. Finally, the quality of the measured quantum individuals is evaluated using the objective function presented in Section 3.3.2 for the APP.

3.4.7. Replacement

At this step, the composition of the quantum population for the next generation is decided. The newly produced population and the old one will be subject to a generational (μ,λ) replacement strategy. In this strategy, the newly produced population entirely replaces the old one and the best quantum individual $\overrightarrow{X_{Q_*}}$ is measured in order to keep its binary state $\overrightarrow{X_{B_*}}$ for the next iteration.

Once the new quantum population is created, a cyclic application of all the above-explained steps is performed until a given stop criterion is met.

4. Experimental results and discussion

In this section, we present the experiments we conducted to practically analyse the real performances of the proposed approach.

4.1. Experimental design

Our experimental design is based on the methodology proposed by Eiben [9] and Talbi [23]. All our tests are carried out using a 2nd generation Intel I3 processor with 2 cores and 4 threads, 2 GB DDR3 Ram and a 1.3 GHz clock speed. All tests were run on a 64-bit windows 7 OS. The implementation was done using a 64-bit version of Matlab 7.12.0 (R2011a), with a Matlab component runtime version 7.15 and a Matlab compiler version 4.15.

To investigate the robustness of the proposed approach, different types of datasets are used: realistic, synthetic and randomly generated. The realistic instance is provided by the University of Malaga representing the city of Malaga, Spain. It was used in several works [10,16,18,22,40,76,78,86] and can be found in [2]. Synthetic instances are provided by the University of Laguna, Spain. They were used in several works [10,21,22,57,58,77] and can be found in [3,4]. Randomly generated instances are used too, they are provided by the University of Constantine II, Algeria. They were used in [88] and available in [1]. Different-sized instances are used in order to assess the scalability of the proposed approach. Instances of dimensions 149, 349, 549, 749 and 1000 are used. Each instance contains a specific number of potential positions of base stations.

Each position is identified by Cartesian coordinates (x, y). The treated area is modelled as a discrete grid where each cell represents a 15 m ∗ 15 m surface. For the case of synthetic instances, the modelled area corresponds to a 18.5 km^2 working area, whereas the random instances correspond to a 20 km^2 working area. Finally, the realistic instance corresponds to a 27 km^2 urban area of Malaga city, Spain (Fig. 14).

Table 2 summarises the used dataset types, their corresponding sizes and the coverage types associated with them. It is worth to mention that *circle* and *omnidirectional* represent the same type of coverage.

Unlike the majority of previous works that used only one type of coverage, we use three types of coverage (see Fig. 2), squared, omnidirectional and directive, introduced in [21]. The coverage parameters and the working area dimensions used here are those used in most literature. They are extracted from [10,20–22,57,58,62,77] for synthetic data. The parameters of realistic data are extracted from [10,16,18,22,40,76,78,86]; and for the randomly generated data, the parameters are extracted from [88]. It is worth to mention that directive antennas cover one-sixth of the area of omnidirectional antennas having the same radius. Table 3 summarises the used antenna coverage parameters.

The proposed QIGA is compared to the state-of-the-art metaheuristics used to solve the APP, which are the PBIL and the GA. They were proposed originally in [71,42], respectively, and used to solve APP in [57,58,76,78,19–21,62,63,65]. The QIGA parameters used in this work have been tuned through several fine-grained experiments. Table 4 summarises the parameters used when evaluating the proposed QIGA and the canonical GA. The PBIL parameters are extracted from [76,78]. It is worth to mention that a *While* loop is used instead of a *For* loop for a fair comparison based on the maximum number of objective function evaluations.

For both the QIGA and the classical GA a roulette wheel selection is used. Also, a two-point crossover is used. The size of the selected quantum parents to create the *mating pool* is set to 50% of the size of the original population.

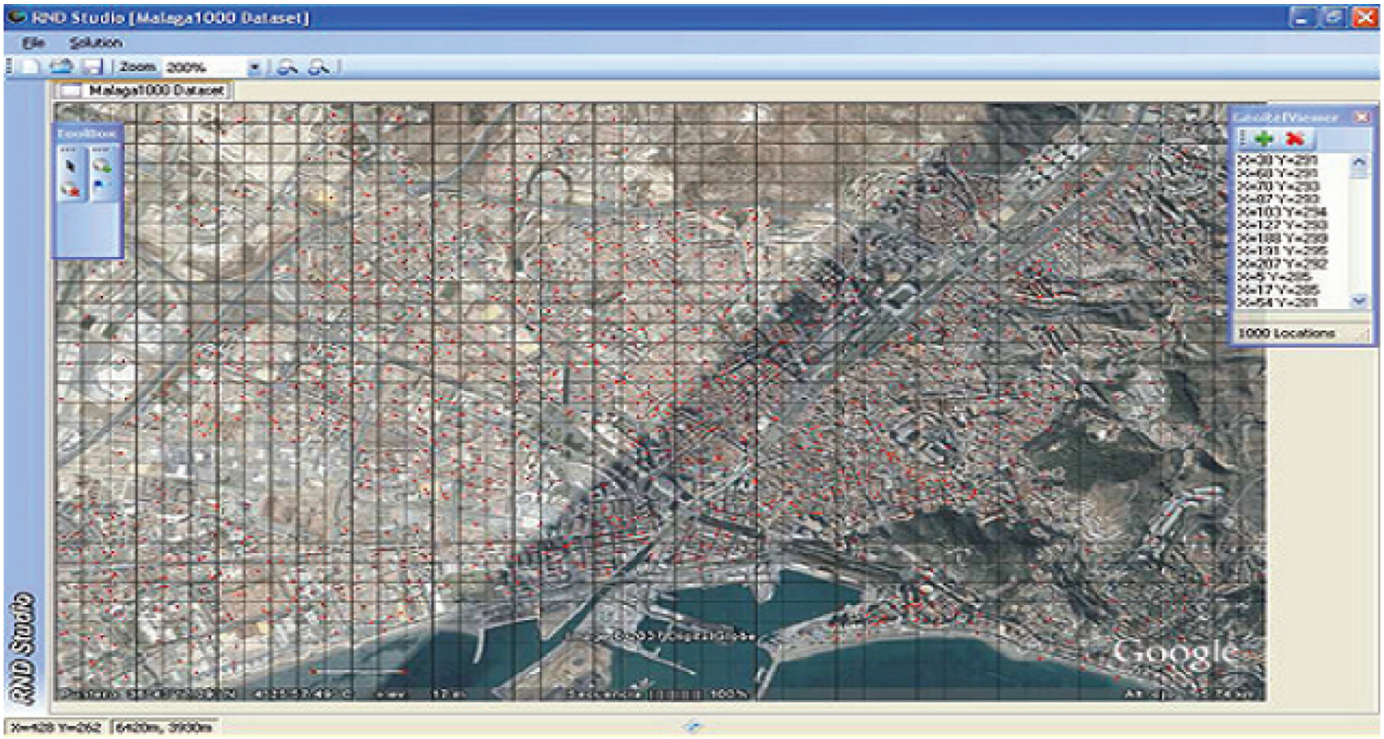


Fig. 14. Discrete representation of the realistic instance: Malaga city, Spain.

Table 2
Instances used: type, size and coverage.

Instance	Grid size	Instance size	Coverage
Synthetic	287×287	149	Omnidirectional Squared Directive
		349	Omnidirectional Squared Directive
Random	300×300	549	Omnidirectional Squared Directive
		749	Omnidirectional Squared Directive
Realistic	450×300	1000	Omnidirectional

Table 3
Antenna coverage parameters.

Synthetic data	Antenna type Radius	Squared 20 cells	Omnidirectional 22 cells	Directive 22 cells
Random data	Antenna type Radius	Squared 24 cells	Omnidirectional 26 cells	Directive 26 cells
Realistic data	Antenna type Radius	Squared None	Omnidirectional 30 cells	Directive None

Table 4
Parameters of the QIGA and classical GA.

Algorithm	Crossover probability (P_c)	Mutation probability (P_m)	Learning rate (LR)
QIGA	0.7	0.05	0.10
GA	0.7	0.05	/

Each algorithm is run till reaching a stop criterion which is chosen to be 100,000 fitness evaluations. 100 individuals are used, so 1000 iterations. All algorithms are repeated for 20 independent runs. Several results are reported such as the *best* and the *worst* fitness values, and also the *mean* and *standard deviation* of fitnesses over 20 independent executions. The technical results obtained by the best individual in each experiment are reported too. The technical results presented are *pure coverage*, *overcoverage*, the *total covered area* and the *number of antennas used*. In order to assess the quality of solutions, several statistical analysis tests such as the *one-sample Kolmogorov–Smirnov test*, the *Bartlett test*, the *Friedman two-way analysis variance test*, the *Kruskal–Wallis one-way* analysis of variance test and a *post hoc test* are performed on the obtained results.

4.2. Experimental results

In this section, we present several numerical results (technical and statistical) obtained by the three algorithms QIGA, PBIL and the GA. Tables 5–9 show the numerical results obtained by the proposed QIGA, the PBIL and the GA for each instance of the problem. On the basis of the metric "*Mean*", the best results are highlighted in bold.

Table 5
Numerical results of the QIGA, PBIL and GA for Instance 149.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
149	Squared	QIGA	188.361	163.792	178.058	6.006
		PBIL	200.466	189.521	**194.208**	3.154
		GA	110.495	99.044	104.332	3.602
	Circle	QIGA	142.326	128.911	137.645	3.159
		PBIL	148.345	140.208	**144.992**	2.301
		GA	97.282	85.472	90.832	2.995
	Directive	QIGA	48.867	48.101	48.518	0.209
		PBIL	49.087	48.431	**48.807**	0.181
		GA	41.543	36.906	38.924	1.101

Table 6
Numerical results of the QIGA, PBIL and GA for Instance 349.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
349	Squared	QIGA	159.492	145.709	152.869	3.897
		PBIL	177.261	152.862	**164.638**	5.803
		GA	63.644	57.858	61.196	1.602
	Circle	QIGA	136.101	124.829	130.298	2.715
		PBIL	144.945	130.543	**136.462**	3.193
		GA	62.199	57.224	59.614	1.491
	Directive	QIGA	52.771	51.741	52.346	0.298
		PBIL	53.717	52.094	**53.058**	0.447
		GA	39.451	37.199	38.304	0.669

Table 7
Numerical results of the QIGA, PBIL and GA for Instance 549.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
549	Squared	QIGA	191.572	174.513	184.026	3.693
		PBIL	195.225	181.509	**187.938**	3.659
		GA	42.777	38.927	41.071	1.195
	Circle	QIGA	169.582	158.262	162.436	2.693
		PBIL	168.995	159.981	**164.694**	2.561
		GA	42.641	39.027	40.621	1.068
	Directive	QIGA	70.023	67.114	68.532	0.688
		PBIL	70.329	67.729	**68.992**	0.621
		GA	37.914	34.444	35.851	0.895

Table 8
Numerical results of the QIGA, PBIL and GA for Instance 749.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
749	Squared	QIGA	192.802	178.257	**186.117**	4.554
		PBIL	186.051	174.676	181.271	3.299
		GA	30.788	28.477	29.437	0.529
	Circle	QIGA	169.157	160.686	**163.651**	2.355
		PBIL	165.888	157.926	161.049	1.777
		GA	30.579	27.929	29.198	0.821
	Directive	QIGA	71.004	68.455	**69.431**	0.708
		PBIL	70.111	67.785	68.815	0.621
		GA	29.608	26.802	27.786	0.665

Table 9
Numerical results of the QIGA, PBIL and GA for Instance 1000.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
1000	Circle	QIGA	143.751	127.559	**137.121**	4.604
		PBIL	138.821	112.869	125.718	5.631
		GA	18.101	16.783	17.403	0.402

Table 10 reports the technical results obtained by the best individuals in each experiment. Technical metrics specific to the APP are reported, where *# Antennas* represents the number of deployed antennas, *Coverage* corresponds to the amount of cells covered only by one antenna and *Overcoverage* is the number of cells covered by more than one antenna. Finally, *Total Coverage* represents the amount of cells covered by one or several antennas. It is worth to mention that the metrics *Coverage*, *Overcoverage* and *Total Coverage* are expressed in terms of grid cells and percentage, respectively.

Table 10
Technical results obtained by the QIGA, PBIL and the GA in Instances 149, 349, 549, 749 and 1000.

Instance	Coverage	Algorithm	# Antennas	Coverage (Cell)	Coverage (%)	Overcoverage (Cell)	Overcoverage (%)	Total coverage (Cell)	Total coverage (%)
149	Squared	QIGA	49	76,509	92.886	2624	3.186	79,133	96.071
		PBIL	49	81,599	99.065	37	0.045	81,636	99.111
		GA	58	41,012	49.791	24,928	30.264	65,940	80.054
	Circle	QIGA	50	64,495	78.301	4990	6.058	69,485	84.358
		PBIL	50	67,256	81.652	3683	4.471	70,939	86.123
		GA	76	41,221	50.044	29,604	35.941	70,825	85.985
	Directive	QIGA	88	41,976	50.961	12,039	14.616	54,015	65.577
		PBIL	90	42,167	51.193	12,581	15.274	54,748	66.467
		GA	89	36,036	43.751	14,049	17.056	50,085	60.806
349	Squared	QIGA	50	66,201	80.371	7355	8.929	73,556	89.301
		PBIL	50	73,054	88.691	4491	5.452	77,545	94.143
		GA	149	15,585	18.921	64,626	78.459	80,211	97.381
	Circle	QIGA	59	63,050	76.546	10,761	13.064	73,811	89.611
		PBIL	55	66,173	80.337	7371	8.949	73,544	89.286
		GA	144	19,418	23.574	58,536	71.066	77,954	94.641
	Directive	QIGA	110	44,909	54.522	17,847	21.667	62,756	76.189
		PBIL	108	45,712	55.497	17,026	20.671	62,738	76.167
		GA	153	31,946	38.784	32,047	38.907	63,993	77.691
549	Squared	QIGA	39	66,078	73.421	11,715	13.017	77,793	86.437
		PBIL	38	68,209	75.788	9309	10.343	77,518	86.131
		GA	232	1867	2.074	87,792	97.547	89,659	99.621
	Circle	QIGA	45	66,219	73.577	12,402	13.781	78,621	87.357
		PBIL	46	67,721	75.246	11,631	12.923	79,352	88.169
		GA	232	3565	3.961	85,950	95.501	89,515	99.461
	Directive	QIGA	87	50,728	56.364	19,518	21.687	70,246	78.051
		PBIL	95	50,794	56.438	22,771	25.301	73,565	81.739
		GA	220	18,344	20.382	63,852	70.947	82,196	91.329
749	Squared	QIGA	37	64,577	71.752	11,438	12.709	76,015	84.461
		PBIL	41	64,268	71.409	14,337	15.931	78,605	87.339
		GA	323	1362	1.513	88,388	98.209	89,750	99.722
	Circle	QIGA	46	66,019	73.354	13,371	14.857	79,390	88.211
		PBIL	45	65,644	72.938	12,116	13.462	77,760	86.401
		GA	326	912	1.013	88,947	98.831	89,859	99.843
	Directive	QIGA	96	50,538	56.153	23,767	26.408	74,305	82.561
		PBIL	96	50,603	56.226	23,233	25.814	73,836	82.041
		GA	314	10,520	11.689	76,259	84.732	86,779	96.421
1000	Circle	QIGA	47	94,802	70.224	16,533	12.247	111,335	82.196
		PBIL	51	87,913	65.121	26,057	19.302	113,970	84.142
		GA	441	6743	4.995	119,960	88.859	126,703	89.343

Tables 11–15 show the running time of the proposed QIGA, the PBIL and the GA over the whole set of executions performed in each problem instance. On the basis of the metric "*Mean*", the best results are highlighted in bold. It is to be noted also that the reported time is measured in seconds.

Table 11
Running time of the QIGA, PBIL and GA for Instance 149.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
149	Squared	QIGA	1009.412	1061.311	**1043.590**	14.477
		PBIL	2435.625	2554.477	2482.573	26.325
		GA	1483.304	1515.515	1500.699	9.278
	Circle	QIGA	1119.180	1232.776	**1160.498**	33.960
		PBIL	2692.140	2901.249	2803.085	52.230
		GA	1535.838	1558.991	1548.818	6.971
	Directive	QIGA	1792.520	1854.077	1822.558	16.950
		PBIL	4464.821	4713.386	4596.702	60.806
		GA	1655.466	1696.502	**1674.262**	11.098

Table 12
Running time of the QIGA, PBIL and GA for Instance 349.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
349	Squared	QIGA	1171.960	1274.785	**1218.438**	29.909
		PBIL	2755.888	3093.299	2932.243	87.314
		GA	3099.463	3150.049	3118.052	13.883
	Circle	QIGA	1312.271	1494.329	**1381.975**	41.064
		PBIL	3257.831	3607.179	3444.376	88.176
		GA	3238.576	3286.463	3268.343	13.334
	Directive	QIGA	2164.649	2307.587	**2256.162**	41.598
		PBIL	5665.527	6000.622	5793.028	82.860
		GA	3402.871	3455.791	3431.772	12.533

Table 13
Running time of the QIGA, PBIL and GA for Instance 549.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
549	Squared	QIGA	1293.004	1465.738	**1352.253**	41.690
		PBIL	3511.180	3808.679	3689.174	97.309
		GA	5517.264	5612.590	5575.412	23.043
	Circle	QIGA	1346.581	1534.329	**1471.110**	43.897
		PBIL	3812.954	4173.805	3993.203	102.904
		GA	5465.000	5579.306	5524.348	26.898
	Directive	QIGA	2291.894	2444.528	**2376.648**	35.669
		PBIL	6100.159	6435.962	6258.478	100.320
		GA	5822.821	5917.550	5864.482	24.923

Table 14
Running time of the QIGA, PBIL and GA for Instance 749.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
749	Squared	QIGA	1450.439	1599.464	**1540.983**	38.001
		PBIL	4673.580	5171.367	4955.010	157.965
		GA	7529.273	7613.798	7569.298	24.939
	Circle	QIGA	1627.743	1747.555	**1683.729**	34.041
		PBIL	4786.335	5664.251	5240.234	221.289
		GA	7576.011	7671.927	7626.660	23.935
	Directive	QIGA	2505.364	2649.511	**2576.805**	38.180
		PBIL	7115.315	7767.886	7334.691	159.148
		GA	7815.907	7915.876	7871.025	26.330

Table 15
Running time of the QIGA, PBIL and GA for Instance 1000.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
1000	Circle	QIGA	4598.770	5061.645	**4798.948**	129.910
		PBIL	10,280.788	11,934.185	11,306.892	387.134
		GA	24,817.186	25,260.933	25,040.343	109.516

Figs. 15–27 show the objective function evolution through generations when using the QIGA, PBIL and GA for each size of problem and for each type of coverage.

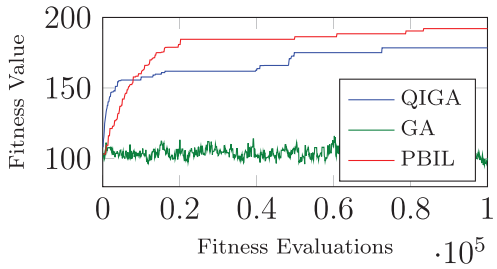


Fig. 15. Instance: 149, coverage: squared.

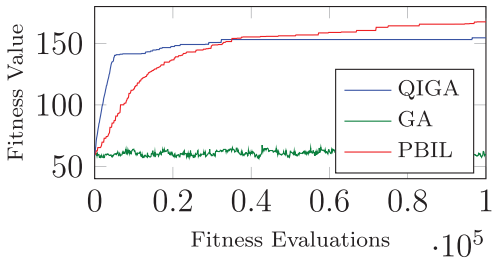


Fig. 16. Instance: 349, coverage: squared.

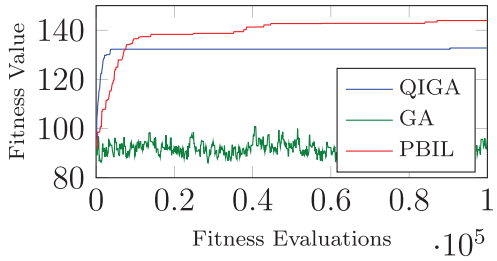


Fig. 17. Instance: 149, coverage: circle.

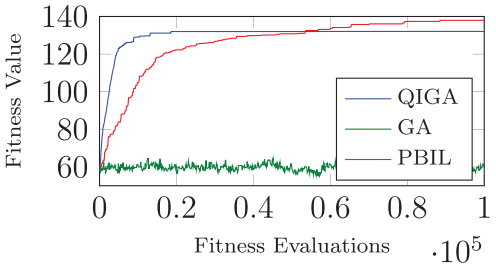


Fig. 18. Instance: 349, coverage: circle.

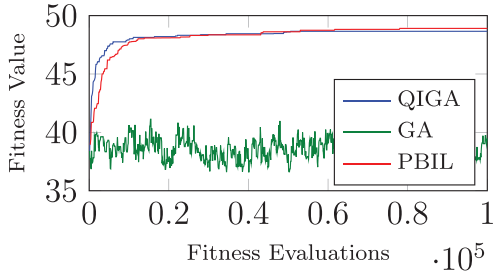


Fig. 19. Instance: 149, coverage: directive.

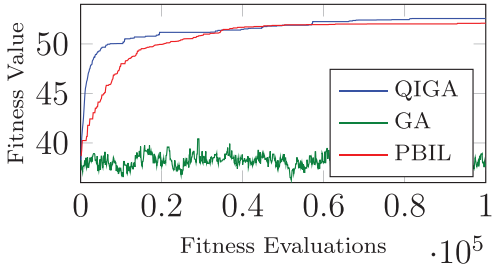


Fig. 20. Instance: 349, coverage: directive.

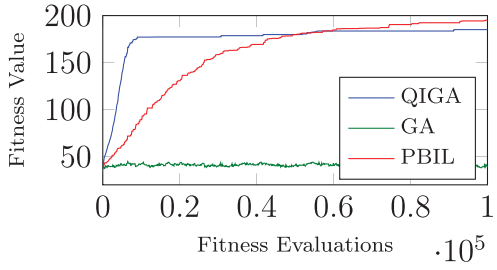


Fig. 21. Instance: 549, coverage: squared.

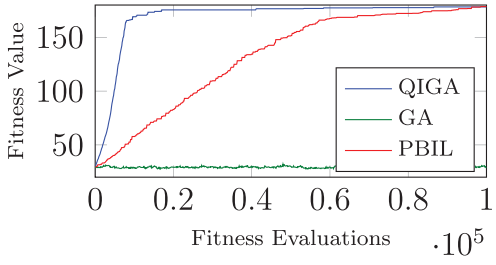


Fig. 22. Instance: 749, coverage: squared.

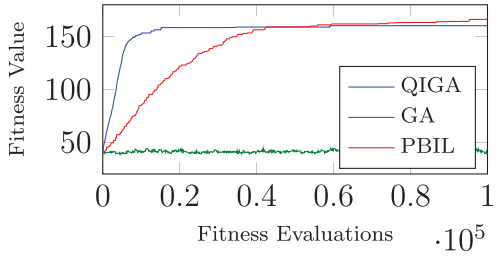


Fig. 23. Instance: 549, coverage: circle.

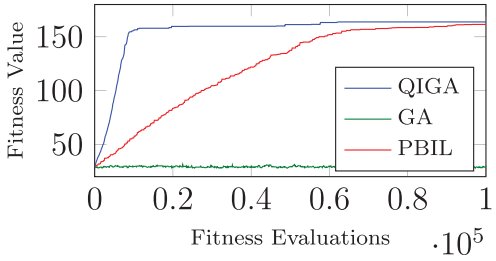


Fig. 24. Instance: 749, coverage: circle.

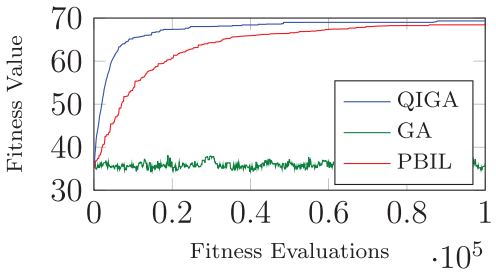


Fig. 25. Instance: 549, coverage: directive.

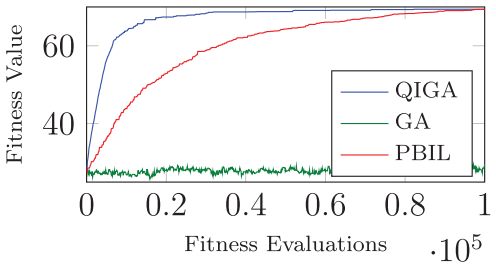


Fig. 26. Instance: 749, coverage: directive.

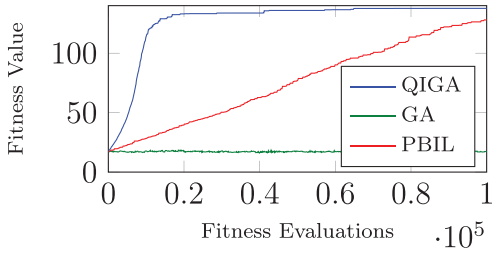


Fig. 27. Instance: 1000.

4.3. Statistical analysis

In this section, we perform a statistical analysis to investigate the performances of each algorithm for each instance. All the tests performed are hypothesis tests. They are defined as follows:

- H_0: defines the null-hypothesis that the assumptions are correct.
- H_1: defines the hypothesis that the assumptions are incorrect.

The assumptions are specific to each test (e.g. distribution normality, variance homogeneity, distribution equality, etc.). They are tests based on the acceptance or rejection of the null-hypothesis H_0. They accept the null-hypothesis if $S_T < C_V$ or $P_V > \varepsilon$. The variable S_T is the value returned by the statistical test, C_V is the critical value used, P_V is the probability value returned by the statistical test and ε is the level of significance.

In order to choose which kind of tests to apply (i.e. parametric or non-parametric), we first apply statistical tests to check two necessary assumptions: normality distribution and variance homogeneity.

We use the one-sample Kolmogorov–Smirnov test as a normality distribution test. Secondly, as homogeneity of variance test, we use the Bartlett test. Finally, on the basis of the results of these tests, we conduct in the next section the Friedman two-way and Kruskal–Wallis one-way tests as distribution equality tests.

4.3.1. Distribution equality tests

In this section, we conduct statistical tests to investigate which algorithm performs better for each instance. For this purpose, we perform first a test to find if there is a difference between *Median* of performance of the algorithms on a specific instance. We use non-parametric tests to find if the distributions of the results obtained by the algorithms are the same or not. If the test fails, this means that at least one algorithm produces results that differ from those obtained by the other algorithms. In this case, a post hoc test has to be done to find what algorithm is different and ultimately deduce which one is the best.

Variance analysis tests are performed on each instance. The four first instances, 149, 349, 549, and 749, include three types of coverage. For this reason, we use the *Friedman two-way analysis of variance test*, since it can simultaneously consider two factors of variability, where the algorithms are *factor A* and the types of coverage are *factor B*.

For the fifth instance, 1000, there is only one coverage used which is the circle one. The Friedman two-way analysis of variance test cannot be used in this case, because there is only one factor to consider which is the algorithm's type (*factor A*). On the basis of this observation, we perform the *Kruskal–Wallis one-way analysis of variance* test.

For both Friedman two-way and Kruskal–Wallis one-way analysis of variance tests, the level of significance used is 5%. In addition, having three algorithms to compare, the degree of freedom used is 2. It is calculated using the formula $(Z-1)$, where Z is the number of algorithms. On the basis of the used level of significance and the degree of freedom, the tests are conducted using a critical value equal to 5.991, according to the *Chi-square probabilities distribution* table. For both tests, the hypotheses are defined as follows:

- H_0: defines the null-hypothesis that the distribution is the same for all the algorithms.
- H_1: defines the hypothesis that the distribution is not the same for all the algorithms.

(A) *Friedman two-way analysis of variance test*: The test is conducted on the results of the 60 independent runs for each algorithm on each instance (i.e. 20 runs for each type of coverage).

As shown in Table 16, for each one of the Instances 149, 349, 549 and 749, the Friedman test succeeds to reject the null-hypothesis H_0. One can conclude that the distribution of the results obtained by the three algorithms is not the same. In this case, a post hoc test is necessary to find which algorithm is different and ultimately deduce which one is better.

Table 16
Friedman two-way variance analysis test for Instances 149, 349, 549 and 749.

Instance	*Pv*	*St*	Null-hypothesis
149	$2.356 \cdot 10^{-33}$	150.257	**Rejected**
349	$9.485 \cdot 10^{-33}$	147.471	**Rejected**
549	$2.766 \cdot 10^{-28}$	126.910	**Rejected**
749	$3.506 \cdot 10^{-29}$	131.041	**Rejected**

(B) *Kruskal–Wallis one-way analysis of variance test*: As shown in Table 17, the test succeeds in rejecting the null-hypothesis H_0. One can conclude that the distribution of the results obtained by the three algorithms is not the same when using Instance 1000. In this case, a post hoc test is also necessary to find which algorithm is different and ultimately deduce which one is better.

Table 17
Kruskal–Wallis one-way analysis of variance test for Instance 1000.

Instance	*Pv*	*St*	Null-hypothesis
1000	$2.000 \cdot 10^{-11}$	49.271	**Rejected**

4.3.2. Post hoc test

A post hoc test is performed to find where the difference occurs and which one of the algorithms is better. In order to find if there is a difference between their distributions or not, the post hoc performs a series of pairwise tests between the two algorithms. It is based on a statistical ranking of algorithms, according to the results obtained by them.

The experiments are performed for 20 executions for each one of the three types of coverage, except for Instance 1000 we have one type of coverage. So, for each size of problem, we performed 60 executions, except for Instance 1000 we performed 20 executions. When performing the post hoc test, we rank the algorithms according to their performances in each one of the 60 executions. Then, the mean of these ranks is taken.

The level of significance used in this test is 5%. The *Pv* obtained by the post hoc test are adjusted using the *Bonferroni-correction procedure*. The hypotheses of the test are defined as follows:

- H_0: defines the null-hypothesis that the distribution is the same for the two algorithms.
- H_1: defines the hypothesis that the distribution is not the same for the two algorithms.

As shown in Table 18, the two algorithms that differ in Instances 149, 349 and 549 are the GA and the PBIL, whereas for Instance 749, it is the QIGA and the GA that differ. When tackling Instance 1000, the three algorithms mutually differ. One can notice that the algorithms PBIL and QIGA do not differ for Instances 149, 349, 549 and 749.

Table 18
Statistical results (*Pv*) of the post hoc test.

Instance	Algorithm	QIGA	PBIL	GA	Null-hypothesis
149	QIGA	/	0.613	/	**Accepted**
	PBIL	/	/	0.019	**Rejected**
	GA	0.425	/	/	**Accepted**
349	QIGA	/	0.675	/	**Accepted**
	PBIL	/	/	0.020	**Rejected**
	GA	0.403	/	/	**Accepted**
549	QIGA	/	1.000	/	**Accepted**
	PBIL	/	/	0.044	**Rejected**
	GA	0.230	/	/	**Accepted**
749	QIGA	/	1.000	/	**Accepted**
	PBIL	/	/	0.267	**Accepted**
	GA	0.037	/	/	**Rejected**
1000	QIGA	/	0.005	/	**Rejected**
	PBIL	/	/	0.000	**Rejected**
	GA	0.000	/	/	**Rejected**

4.4. Discussion

On the basis of the "*Mean*" metric given in Tables 5–9, one can see that for Instances 149, 349 (squared and circle coverages), the PBIL slightly outperforms the proposed QIGA, whereas for Instance 549, the QIGA achieves results as good as those obtained by the PBIL algorithm. In addition, on the basis of the "*Best*" metric, the proposed QIGA outperforms the PBIL in Instance 349 (circle coverage). For higher instances: 749 and 1000, one can see that the proposed QIGA outperforms the PBIL algorithm for all types of coverage. It is also worth to mention that for Instances 149, 349, 549, 749 and 1000, and for all types of coverage, the GA is the algorithm that gives the worst results.

On the basis of the "*STD*" metric in Tables 5–9, we can see that for Instance 149 (squared and circle coverages) the proposed QIGA has the largest deviation. In addition, one can observe that as the problem size increases (Instances 349, 549, 749 and 1000) the deviation of the proposed QIGA decreases and becomes more or less similar to the one obtained by the PBIL algorithm. It is worth to mention that for all sizes of the problem (squared and circle coverages) the GA has the smallest standard deviation among all algorithms, whereas for the circle coverage in all sizes of problem instance, the three algorithms PBIL, QIGA and GA have all similar standard deviation.

According to the "*Mean*" and "*STD*" metrics in Tables 5–9, one can observe that the proposed QIGA is barely less efficient than the PBIL because when tackling small instances (149 and 349) the QIGA tends to have a slower convergence than that of the PBIL, whereas for higher instances (Instances 549, 749 and 1000) the QIGA balance between exploitation and exploration (i.e. convergence and divergence) becomes more stable, which explains the fact that the proposed QIGA outperforms the PBIL algorithm.

With regard to fitness evolution of the proposed QIGA, PBIL and GA displayed in Figs. 15–27, one can see that for Instances 149 and 349 (squared and circle coverages) the QIGA has a slower convergence than the PBIL algorithm, whereas for circle coverage, both algorithms gave similar convergence rate. Taking now higher instances: 349, 549, 749, 1000 and for all types of coverage (squared, circle and directive), we can see that the convergence rate of the QIGA increases as the size of the problem increases, while the convergence rate of the PBIL decreases as the size of the problem increases. These findings confirm the conclusion made above about the efficiency of the proposed QIGA. Also, they support the fact that the scalability of the proposed QIGA is superior to the PBIL and GA. In fact, the size's problem mainly influences the QIGA convergence rate.

On the basis of the technical results shown in Table 10, one can say that for Instances 149 and 349 (squared and circle coverages) the PBIL barely outperforms the proposed QIGA, while for directive coverage, the proposed QIGA gives results as good as the ones obtained by the PBIL algorithm. Considering Instance 549, one can see that the proposed QIGA obtains results as good as the ones achieved by the PBIL, whereas for the case of directive coverage, the proposed QIGA outperforms the PBIL algorithm. For higher instances (749 and 1000) and for all types of coverage, the proposed QIGA outperforms the state-of-the-art PBIL algorithm. It is also worth to mention that the proposed QIGA is able to outperform the GA in all problem sizes and for all coverage types.

With respect to the running time displayed in Tables 11–15, one can see that the proposed QIGA outperforms both the PBIL and GA in all problem instances. In fact, in 13 out of 13 instances, the QIGA has a shorter running time than the one of PBIL, while in 12 out of 13 instances, the QIGA achieves a running time better than the one of the GA.

Finally, as seen from the results of the statistical post hoc test shown in Table 18, one can note that for Instances 149, 349 and 549, the two algorithms that statistically differ are the PBIL and the GA, whereas for Instance 749, the QIGA and the GA are the ones that differ. For Instance 1000, every algorithm differs from the others. On the basis of these findings, one can see that no statistical difference was found between the proposed QIGA and the PBIL algorithm for Instances 149, 349 and 549, which means that the proposed QIGA produces results as good as the PBIL algorithm, while for Instances 749 and 1000, a statistical difference is found between both algorithms, which means that the QIGA outperforms the PBIL. Taking now the GA, it is worth to mention that the latter was statistically outperformed in all the problem instances by our proposed QIGA.

4.5. Parameter sensitivity analysis: the learning rate

For further understanding of the influence of the Learning Rate (*LR*) value on the behaviour of the proposed approach, experiments on all instances of the APP are run with several successive values of *LR*.

Since *LR* is drawn from the interval [0,1], values 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1 are used. In addition, on the basis of the tuning performed for P_c and P_m, for all experiments, values of crossover and mutation probabilities are set to 0.7 and 0.05, respectively. Besides, for consistency and scientific coherence, we use the same experimental parameters used in our main experimental study. Thus, within the sensitivity analysis experiments, the proposed approach is run until reaching the stop criterion of 100,000 fitness evaluations. The quantum population is composed of 100 quantum individuals and run for 1000 iterations. Finally, each experiment is repeated for 20 independent runs.

On the basis of the results obtained in these experiments, one can deduce that the *LR* value affects the efficiency of the proposed QIGA. Indeed, for Instances 149 and 349 (squared and circle coverages), higher values of *LR* ($LR \geq 0.4$) produce better results than smaller values. For Instance 549 (squared and circle coverages), all values have similar results. For higher instances: 749 and 1000 (squared and circle coverages), the influence of *LR* value is in the opposite than when tackling small instances. In fact, these results show that smaller values of *LR* ($LR \leq 0.4$) produce better results than bigger values. It is also worth to mention that for all sizes of instances, when tackling the directive coverage all values of *LR* produce similar results. In addition, the experiments show that the worst results are obtained when setting *LR* to 0, and this for all types or sizes of problem instances.

Besides changes in the efficiency, a change also in the behaviour can be noticed when using different values of *LR*. Indeed, the results show that for Instances 149 and 349, small values of *LR* ($LR \leq 0.4$) induce a slower convergence rate of the proposed approach, whereas higher values of *LR* produce a quicker convergence rate. For Instance 549, the difference in the behaviour induced by different values of *LR* is less noticeable, since all values produce more or less similar behaviour. Finally, for higher instances: 749 and 1000, we can observe that smaller values of *LR* ($LR \leq 0.4$) produce quicker convergence rate, unlike bigger values that result in slower convergence rate. Figs. 28 and 29 illustrate the produced behaviour when tackling Instances 349 and 1000, respectively.

On the basis of the results obtained in the above-presented experiments, one can also note that when setting *LR* to 0, the proposed approach produces much worst results than when setting *LR* to 1 or other non-null values. This can be mathematically explained by the fact that when $LR=0$, Formula (8) describing the novel learning gate becomes $\alpha' = \frac{XB_*}{1} = XB_*$. So, in this case the *Q-bit* will take the value of

the corresponding bit in the binary state (i.e. solution) $\overrightarrow{X_{B_*}}$ produced by the best quantum individual $\overrightarrow{X_{Q_*}}$. Thus, this will bring the proposed approach to the loss of state superposition. In such context, the proposed QIGA becomes a classical binary-coded GA. However, when setting *LR* to a value different from 0, Formula (8) remains mathematically unchanged, which prevents the loss of state superposition. Figs. 28 and 29 confirm the deduced conclusions.

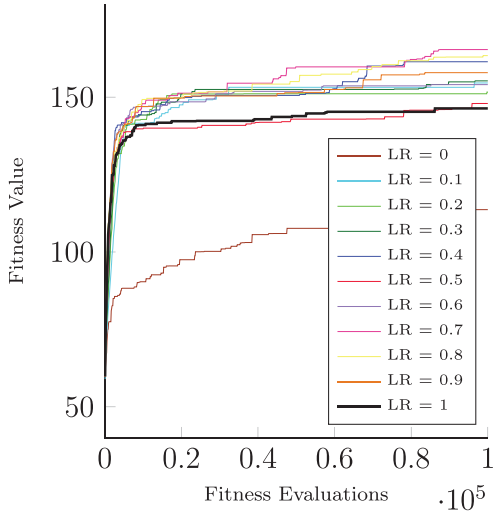


Fig. 28. Instance: 349, coverage: squared.

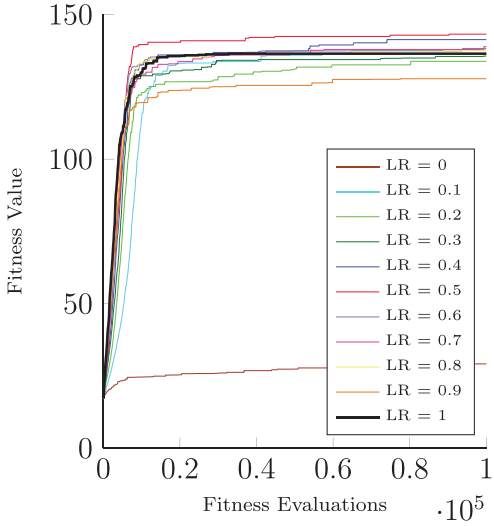


Fig. 29. Instance: 1000, coverage: circle.

5. Conclusion

In this paper, a novel QIGA based on a new quantum gate has been presented to solve an NP-hard binary optimisation problem widely faced in the field of advanced cellular phone networks: the APP.

In order to assess the scalability, the efficiency and robustness of the proposed approach, many experiments have been carried out on realistic, synthetic and random data with different dimensions. Several statistical analysis tests have been conducted to assess the quality of solutions. The state-of-the-art algorithms, the PBIL and the GA, previously designed to solve the APP, were taken as comparison basis.

The results obtained from the experiments show that the proposed approach is robust and efficient when tackling the APP. In fact, the proposed approach outperformed the state-of-the-art algorithm PBIL for the six most complex instances. Besides, the numerical results showed that the proposed QIGA gave results as good as those obtained by the PBIL algorithm in the remaining instances. Taking now the GA, experiments showed that it was outperformed by our proposed QIGA in 13 out of 13 instances.

As perspective of research, we intend to apply the proposed approach on more complex and dynamic real-world APP instances. We also think about deploying the proposed algorithm within a prototype software and test it with more real-world scenarios and using different generation of cellular networks (2G, EDGE, GPRS, 3G, $3G^+$, LTE and 4G). Finally, we are working on the integration of our approach within the frameworks Network Simulator 2 and 3 (NS 2 and 3) using realistic data from OpenStreetMap platform.

Appendices description

For more insight and understanding of the paper, a set of additional detailed materials is organised as appendices. Firstly, a more detailed description of the proposed approach and also the algorithms taken as a comparison basis, the PBIL and the GA, is given. Then, we present a description of the methodology followed within our statistical study. Finally, we present materials about the parameters tuning tables, the figures and tables of the statistical analysis tests performed (e.g. the one-sample Kolmogorov–Smirnov test, the Bartlett test and the post hoc test).

Appendix A. The proposed approach

Pseudo-code of Algorithm 2 gives a detailed description of the implementation of the proposed approach.

Algorithm 2. The proposed approach.

```
---------- Parameters Initialisation ----------
Set N the number of quantum individuals XQ in the quantum population Q-pop.
Set d the length of every XQ in Q-pop (d is equal to the number of available locations in the network).
Set P_m the quantum mutation probability, P_m ∈ [0,1].
Set P_c the quantum crossover probability, P_c ∈ [0,1].
Set LR the learning rate of the learning gate, LR ∈ [0,1].
---------- Initialisation ----------
for i =1:N (For every quantum individual XQ in the quantum population Q-pop) do
    for j =1:d (For every Q-bit in the quantum individual XQ) do
        Set α_ij to 1/√2.
        Set β_ij to 1/√2.
    end for
end for
for i =1:N (For every quantum individual XQ in the quantum population Q-pop) do
    Compute the binary state XB_i by measuring the quantum individual XQ_i using Formula (7).   % (see Line 51)
    Compute f_i the fitness value of the binary state XB_i using Formula (9).   % (see Line 54)
end for
Extract the best quantum individual XQ_* and its corresponding binary state XB_*.
---------- Search Process ----------
while stop criterion not reached do
    ---------- Selection: Roulette Wheel ----------
    for i =1:N (For every quantum individual XQ in the quantum population Q-pop) do
        Compute p_i = f_i / Σ_{i=1}^{N} f_i.
        Compute θ_i = p_i * 2π.
        Create a disk where the sector representing each quantum individual XQ_i is proportional to its angle θ_i.
    end for
    for i =1:(N/2) do
        Generate rand_angle a random angle from the interval [0,2π] according to a uniform distribution.
        Add to the mating pool the quantum individual whose sector includes rand_angle.
    end for
    ---------- Quantum Crossover: Quantum Two-Point Crossover ----------
    for i=1:(N/4) do
        Create the i^th couple by randomly selecting two quantum parents from the mating pool.
        Generate a random number rand from a standard uniform distribution U(0,1).
        if rand ⩽ P_c then
            Apply a quantum two-point crossover by generating two random numbers rand1 and rand2 from the
            interval [1,d] according to a uniform distribution.
            Swap the quantum parents of the i^th couple at the switching points rand1 and rand2.
        else
            Copy the two quantum parents of the i^th couple and consider them as new quantum offspring.
        end if
    end for
    ---------- Quantum Mutation: Quantum Bit-Flip Mutation ----------
    for i=1:(N/2) (For all (N/2) newly-produced quantum offspring XQ_off) do
        for j=1:d (For every Q-bit in the newly-produced quantum offspring XQ_off) do
            Generate a random number rand from a standard uniform distribution U(0,1).
            if rand ⩽ P_m then
                Apply a quantum bit-flip mutation by inverting probabilities α and β in the j^th Q-bit.
            else
                Keep the Q-bit as it is.
            end if
        end for
    end for
    ---------- Quantum Interference: Learning Gate ----------
    for i=1:N (For (N/2) newly-produced quantum offspring and (N/2) parents) do
        Perform interference on the quantum individual using Formula (8) of the learning gate.
    end for
    ---------- Quantum Measurement ----------
    for i=1:N (For (N/2) newly-produced quantum offspring and (N/2) parents) do
        Compute the binary state XB_i by measuring the quantum individual XQ_off i or XQ_Par i using Formula (7).
    end for
    ---------- Evaluation ----------
    for i=1:N (For all the (N) measured quantum individuals do)
        Evaluate the goodness of the produced quantum offspring XQ_Off i and quantum parent XQ_Par i by evaluating
        its binary state XB_i using Formula (9).
    end for
    ---------- Replacement: Generational (μ,λ) Strategy ----------
    Replace the previous quantum population Q-pop by the newly-produced mating pool.
    Choose the best quantum individual from the union of the previous quantum population Q-pop and the newly
    -produced mating pool as the new best quantum individual XQ_* for the population of the next generation.
end while
```

Appendix B. The population-based incremental learning and genetic algorithms

As stated in Section 4.1, the PBIL and the GA are taken as a comparison basis within our experimental study. They were proposed originally in [71,42], respectively, and used to solve the APP in [57,58,76,78,19–21,62,63,65]. In this section, we give a brief overview of both algorithms. Their main frameworks are summarised by pseudo-codes of Algorithms 3 and 4, respectively.

Algorithm 3. PBIL pseudo-code.

1: Initialisation of probability vector.
2: **Repeat**
3: Generate population.
4: Evaluation.
5: Update the probability vector.
6: Mutate the probability vector.
7: **Until** Stop criterion is reached

Algorithm 4. GA pseudo-code.

1: Initialisation of population.
2: **Repeat**
3: Selection.
4: Variation operators.
5: Evaluation.
6: Replacement.
7: **Until** Stop criterion is reached

In the following, we give further insight into both the PBIL and GA. For more details, one can refer to original works of Baluja [71] and Holland [42]. It is also worth to mention that d in the next sections refers to the size of the problem being tackled.

B.1. Population-based incremental learning algorithm

The PBIL algorithm was first proposed by Baluja in [71]. It was born from the combination of concepts from evolutionary computation and competitive learning from artificial neural networks. The main idea of the PBIL is to evolve a probability vector $\overrightarrow{Pv}$ from which every individual in the population is drawn at each iteration.

As shown in Algorithm 3, the PBIL starts by *initialising* the probability vector $\overrightarrow{Pv} = \{Pv_1, Pv_2, \ldots, Pv_d\}$. Each element Pv_j of this vector, where $j = 1\ldots d$, is initially set to the same value 0.5. Then, like every evolutionary algorithm does, the PBIL performs a series of variation operators that make the probability vector $\overrightarrow{Pv}$ evolve toward a fitter state.

Firstly, it *generates* N individuals $\overrightarrow{X}$, where each individual $\overrightarrow{X_i} = \{X_{i,1}, X_{i,2}, \ldots, X_{i,d}\}$, and $i = 1\ldots N$. This is done by generating N vectors of random numbers $\overrightarrow{R}$ drawn from a uniform standard distribution, where each $\overrightarrow{R_i} = \{R_{i,1}, R_{i,2}, \ldots, R_{i,d}\}$. Each individual $\overrightarrow{X}$ is drawn by affecting either the value 1 to the dimension X_{ij}, if $R_{ij} \leq Pv_j$, otherwise it will take the value 0. Secondly, each generated individual $\overrightarrow{X_i}$ is *evaluated* using the objective function of the problem being addressed. Then, the best individual $\overrightarrow{X_*}$ is extracted. Thirdly, the probability vector $\overrightarrow{Pv}$ is updated using Formula (B.1), where $LF \in [0,1]$ is the learning factor:

$$Pv_j = Pv_{j*}(1.0 - LF) + X_{*j*}LF \tag{B.1}$$

Finally, the probability vector $\overrightarrow{Pv}$ is *mutated* by generating a vector $\overrightarrow{R}$ of random numbers drawn from a standard uniform distribution, where $\overrightarrow{R} = \{R_1, R_2, \ldots, R_d\}$. Then, if $R_j \leq P_m$, the new value of Pv_j is computed using Formula (B.2), where $P_m \in [0,1]$ and $M_A \in [0,1]$ represent the mutation probability and the mutation amount, respectively.

$$Pv_j = Pv_{j*}(1.0 - M_A) + R_{j*}M_A \tag{B.2}$$

B.2. Genetic algorithm

The GA was first proposed by Holland in [42]. It was born from the attempt to mimic the natural process of reproduction and evolution. The idea of the GA is to evolve a set of individuals toward a fitter state by applying a series of selection, variation, evaluation and replacement phases. As shown in Algorithm 4, the GA starts by *initialising* a population of N individuals $\overrightarrow{X}$, where each individual $\overrightarrow{X_i} = \{X_{i,1}, X_{i,2}, \ldots, X_{i,d}\}$, and $i = 1\ldots N$ and $j = 1\ldots d$. The latter are evaluated using the fitness function of the problem being addressed, then the best individual $\overrightarrow{X_*}$ is extracted.

In each iteration of the algorithm, firstly M individual parents $\overrightarrow{P}$ are selected according to a given strategy. Secondly, the selected parents are perturbed using *variation* operators which are the mutation and crossover; in order to produce $(N-M)$ new offspring $\overrightarrow{O}$. The crossover consists of exchanging parts of the parent individuals at some chosen points. The amount of information exchanged and the number of crossover operations performed are ruled by both the type of crossover used and the crossover probability $P_c \in [0, 1]$. Then, the mutation operator randomly mutates the values of genes in each produced offspring $\overrightarrow{O_k}$, where $k = 1\ldots(N - M)$. The number of mutation operations is determined by a mutation probability $P_m \in [0, 1]$. Later, the produced offspring are *evaluated* using the objective function of the problem being addressed.

Finally, having the old population of parents and the newly produced offspring, a replacement phase is performed in order to determine what will be the composition of the population for the next iteration. However, it is complex to draw the features of the basic GA, since generally its mechanics are left largely as a black box due to the existence of several types of selection, crossover, mutation and replacement.

Appendix C. Statistical methodology

Pseudo-code of Algorithm 5 gives a detailed description of the methodology followed within the statistical analysis performed in the paper. Since statistical analysis is performed on a sample of data, it is worth to mention that when referring to the algorithms we point to the results they achieved.

Algorithm 5. The statistical analysis methodology.

```
Perform the one-sample kolmogorov-Smirnov test to check the distribution normality.
if One-Sample kolmogorov-Smirnov Test Succeeds (Normal Distribution) then
  | Perform the Bartlett test to check the homogeneity of variances.
  | if Bartlett Test Succeeds (Homogeneous Variances) then
  | | Perform parametric test: ANOVA tests to check distribution equality.
  | | if Parametric Test Succeeds (Equal Distributions) then
  | | | The distributions are the same and the algorithms performances are equal.
  | | else{% Parametric Test Fails (Non-Equal Distributions)}
  | | | Perform a post-hoc test.
  | | | Find where the differences occur and deduce which algorithm is the best.
  | | end if
  | else{% Bartlett Test Fails (Non-homogeneous variances)}
  | | Perform non-parametric test: Friedman, Kruskal-Wallis to check distribution equality.
  | | if Non-parametric Test Succeeds (Equal Distributions) then
  | | | The distributions are the same and the algorithms performances are equal.
  | | else{% Non-parametric Test Fails (Non-Equal Distributions)}
  | | | Perform a post-hoc test.
  | | | Adjust the Pval using Bonferroni-correction procedure.
  | | | Find where the differences occur and deduce which algorithm is the best.
  | | end if
  | end if
else{% One-Sample kolmogorov-Smirnov Test Fails (Non-Normal Distribution)}
  | Perform non-parametric test: Wilcoxon Rank-Sum, Friedman, Kruskal-Wallis.
  | if Non-parametric Test Succeeds (Equal Distributions) then
  | | The distributions are the same and the algorithms performances are equal.
  | else{% Non-parametric Test Fails (Non-Equal Distributions)}
  | | Perform a post-hoc test.
  | | Adjust the Pval using Bonferroni-correction procedure.
  | | Find where the differences occur and deduce which algorithm is the best.
  | end if
end if
```

All the tests performed within our experimental study are hypothesis tests. They are defined as follows:

- H_0: defines the null-hypothesis that the assumptions are correct.
- H_1: defines the hypothesis that the assumptions are incorrect.

The assumptions are specific to each test (e.g. distribution normality, variance homogeneity, distribution equality, etc.). They are tests based on the acceptance or rejection of the null-hypothesis H_0. They accept the null-hypothesis if $S_T < C_V$ or $P_V > \varepsilon$. The variable S_T is the value returned by the statistical test, C_V is the critical value used, P_V is the probability value returned by the statistical test and ε is the level of significance.

Appendix D. The one-sample Kolmogorov–Smirnov test for normality distribution

Before deciding which type of statistical analysis tests we will apply (i.e parametric or non-parametric), two assumptions have to be verified. The first assumption is the *distribution normality*.

For this purpose we apply the one-sample Kolmogorov–Smirnov test. It is used to test if the results obtained by the algorithms follow a normal (i.e. Gaussian) distribution or not. The results of this test are necessary to decide whether apply parametric or non-parametric statistical analysis tests.

The level of significance is set to 5%. The one-sample Kolmogorov–Smirnov test is conducted using a critical value equal to 0.2941 according to the *Kolmogorov–Smirnov table*. The hypotheses of this test are defined as follows:

- H_0: defines the null-hypothesis that the data follow a normal distribution.
- H_1: defines the hypothesis that the data do not follow a normal distribution.

As shown in Tables D1–D5, for each algorithm and for each problem instance, the one-sample Kolmogorov–Smirnov test fails to reject the null-hypothesis H_0. This confirms that all the data follow a normally distributed function.

Table D1
The one-sample Kolmogorov–Smirnov test for Instance 149.

Instance	Coverage	Algorithm	*Pv*	*St*	Null-Hypothesis
149	Squared	QIGA	0.646	0.158	**Accepted**
		PBIL	0.144	0.248	**Accepted**
		GA	0.968	0.103	**Accepted**
	Circle	QIGA	0.772	0.141	**Accepted**
		PBIL	0.986	0.095	**Accepted**
		GA	0.795	0.138	**Accepted**
	Directive	QIGA	0.249	0.220	**Accepted**
		PBIL	0.482	0.180	**Accepted**
		GA	0.951	0.109	**Accepted**

Table D2
The one-sample Kolmogorov–Smirnov test for Instance 349.

Instance	Coverage	Algorithm	*Pv*	*St*	Null-Hypothesis
349	Squared	QIGA	0.947	0.110	**Accepted**
		PBIL	0.999	0.067	**Accepted**
		GA	0.757	0.143	**Accepted**
	Circle	QIGA	0.823	0.134	**Accepted**
		PBIL	0.910	0.119	**Accepted**
		GA	0.843	0.130	**Accepted**
	Directive	QIGA	0.820	0.134	**Accepted**
		PBIL	0.533	0.173	**Accepted**
		GA	0.903	0.120	**Accepted**

Table D3
The one-sample Kolmogorov–Smirnov test for Instance 549.

Instance	Coverage	Algorithm	*Pv*	*St*	Null-Hypothesis
549	Squared	QIGA	0.620	0.161	**Accepted**
		PBIL	0.839	0.131	**Accepted**
		GA	0.781	0.140	**Accepted**
	Circle	QIGA	0.655	0.157	**Accepted**
		PBIL	0.962	0.105	**Accepted**
		GA	0.705	0.150	**Accepted**
	Directive	QIGA	0.990	0.092	**Accepted**
		PBIL	0.880	0.124	**Accepted**
		GA	0.947	0.110	**Accepted**

Table D4
The one-sample Kolmogorov–Smirnov test for Instance 749.

Instance	Coverage	Algorithm	*Pv*	*St*	Null-Hypothesis
749	Squared	QIGA	0.675	0.154	**Accepted**
		PBIL	0.760	0.143	**Accepted**
		GA	0.997	0.083	**Accepted**
	Circle	QIGA	0.921	0.116	**Accepted**
		PBIL	0.451	0.185	**Accepted**
		GA	0.950	0.109	**Accepted**
	Directive	QIGA	0.914	0.118	**Accepted**
		PBIL	0.999	0.071	**Accepted**
		GA	0.699	0.151	**Accepted**

Table D5
The one-sample Kolmogorov–Smirnov test for Instance 1000.

Instance	Coverage	Algorithm	*Pv*	*St*	Null-Hypothesis
1000	Circle	QIGA	0.826	0.133	**Accepted**
		PBIL	0.960	0.106	**Accepted**
		GA	0.962	0.106	**Accepted**

Mathematically speaking, the one-sample Kolmogorov–Smirnov test is used to test if a given data follows a standard normal distribution with mean 0 and standard deviation equal to 1 ($N(0,1)$). For the case of the APP, results of a given instance are scaled to this range using the formula $\left(\frac{data - Mean}{Standard\ Deviation}\right)$. It is worth to mention that the *mean* and *standard deviation* used to scale the data in this test are the metrics "*Mean*" and "*STD*" presented in Tables 5–9 in the main paper.

The one-sample Kolmogorov–Smirnov is based on the difference between the Cumulative Distribution Function (CDF) of a given data and the one of the theoretic standard normal distribution. Figs. D1–D13 show the similarity and the difference between the two distributions.

Note that *CDF* stands for Cumulative Distribution Function and *SN* stands for Standard Normal.

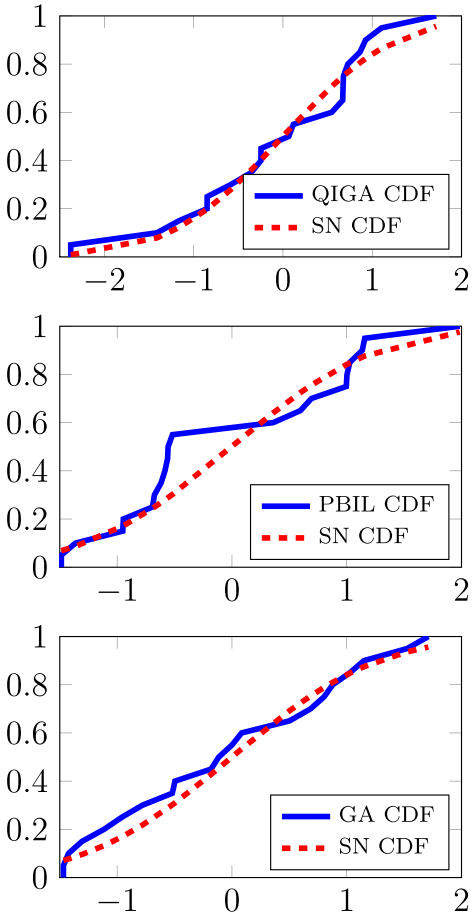


Fig. D1. Instance: 149, coverage: squared.

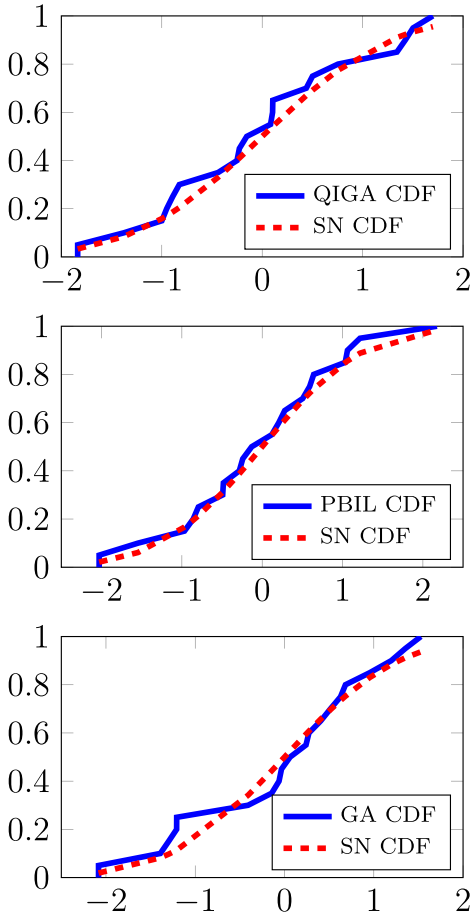
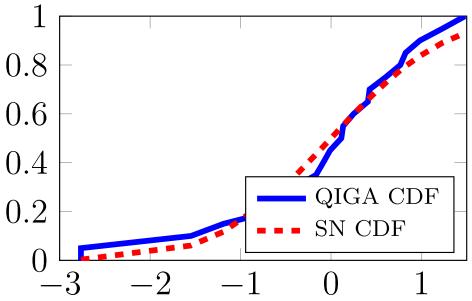
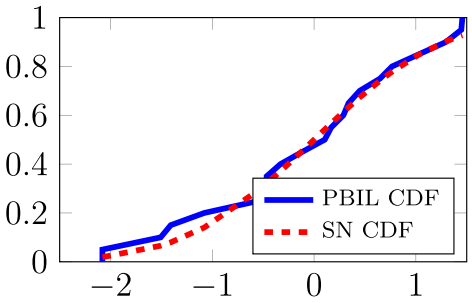


Fig. D2. Instance: 349, coverage: squared.





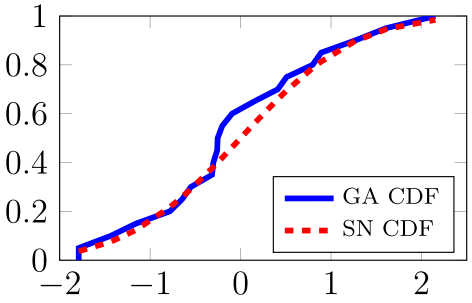
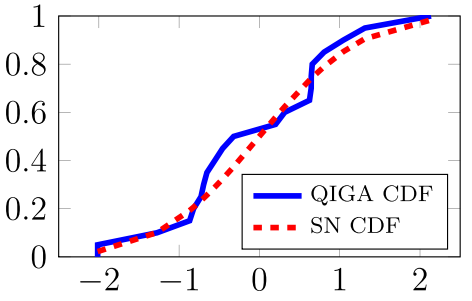
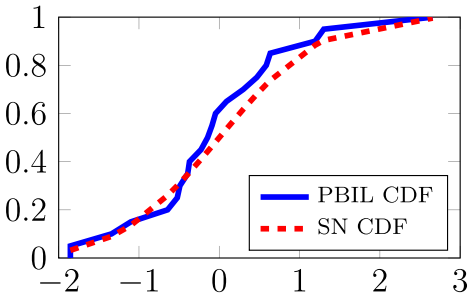


Fig. D3. Instance: 149, coverage: circle.





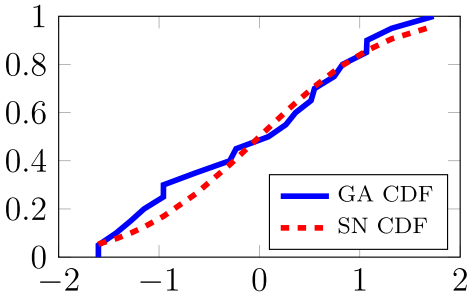
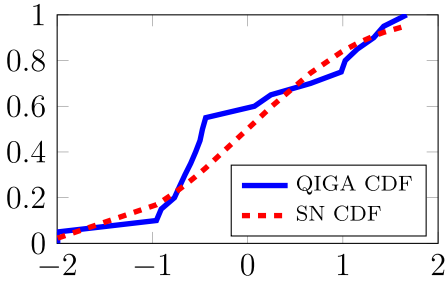
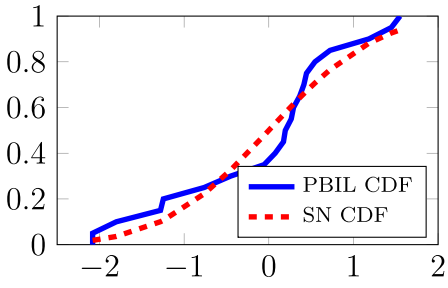


Fig. D4. Instance: 349, coverage: circle.





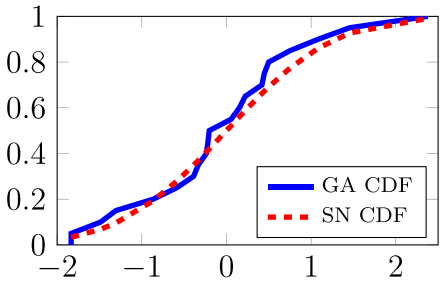
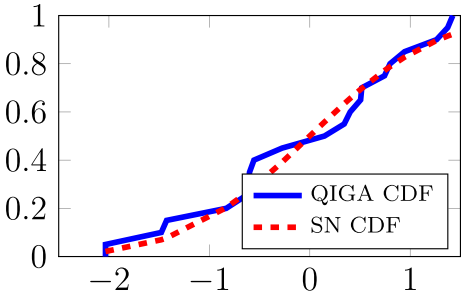
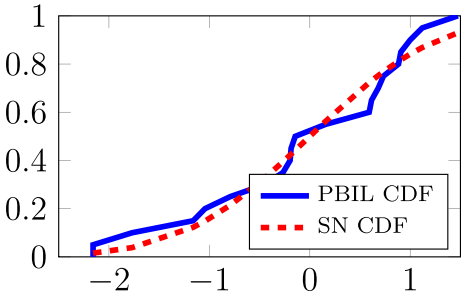


Fig. D5. Instance: 149, coverage: directive.





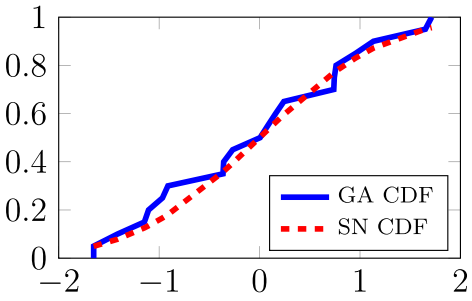
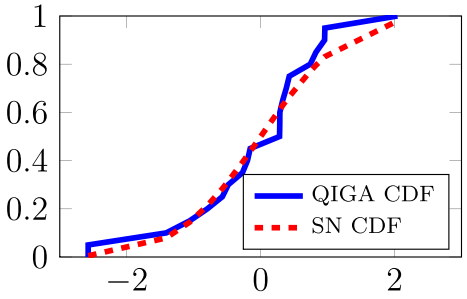
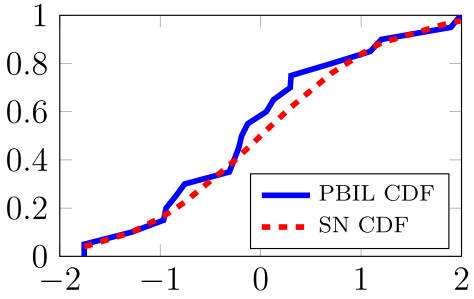


Fig. D6. Instance: 349, coverage: directive.





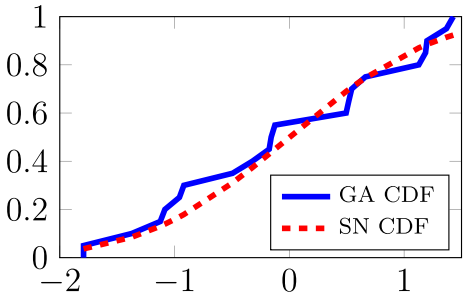
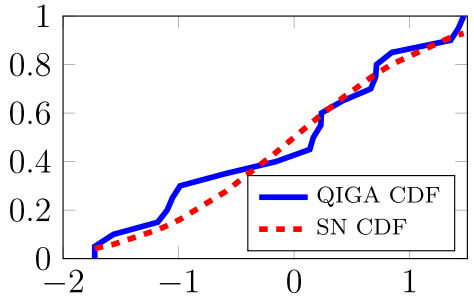
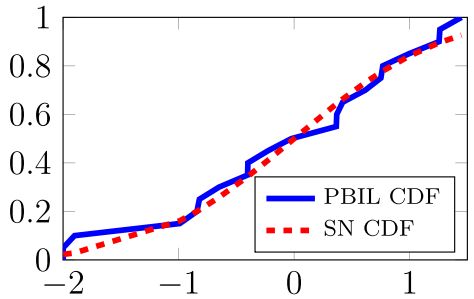


Fig. D7. Instance: 549, coverage: squared.





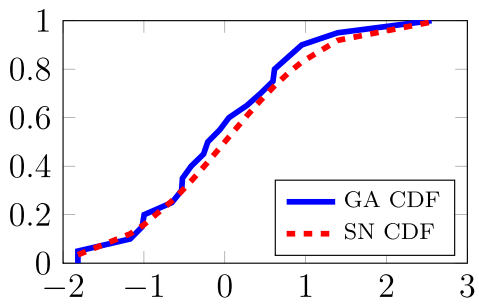
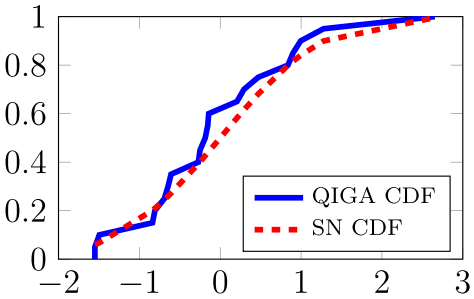
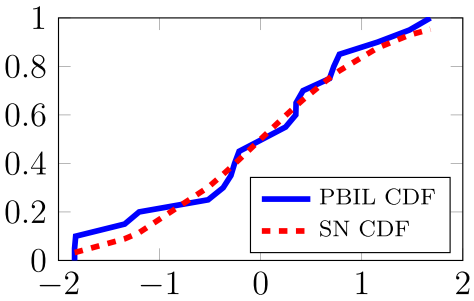


Fig. D8. Instance: 749, coverage: squared.





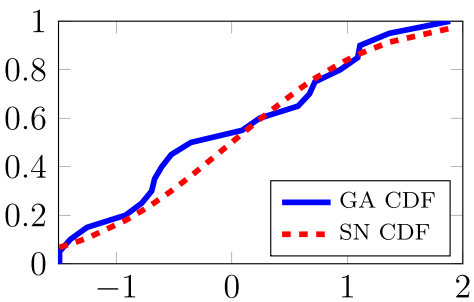
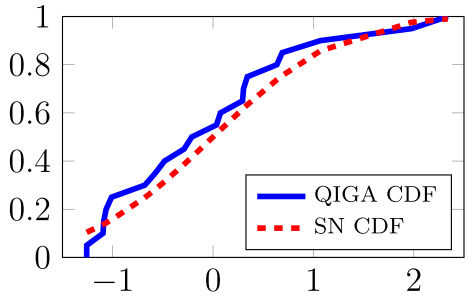
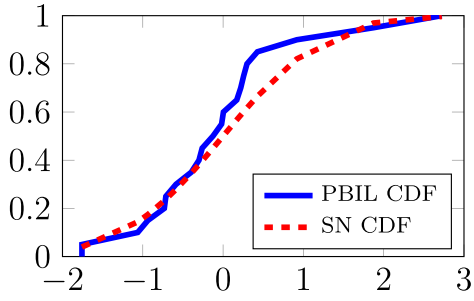


Fig. D9. Instance: 549, coverage: circle.





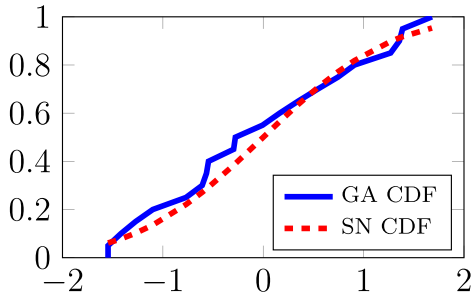


Fig. D10. Instance: 749, coverage: circle.

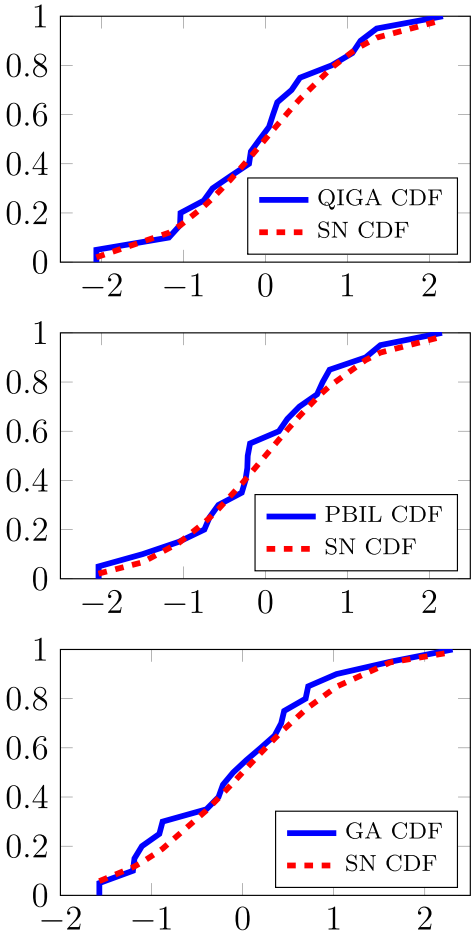


Fig. D11. Instance: 549, coverage: directive.

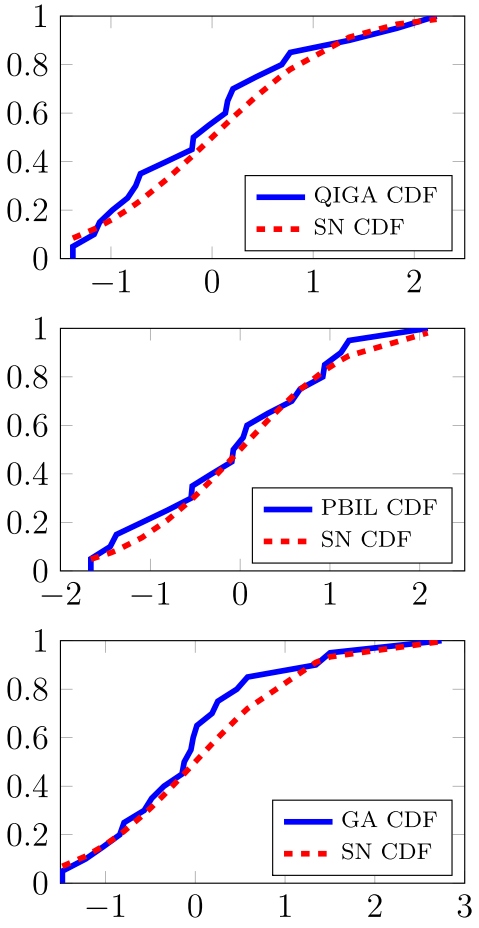


Fig. D12. Instance: 749, coverage: directive.

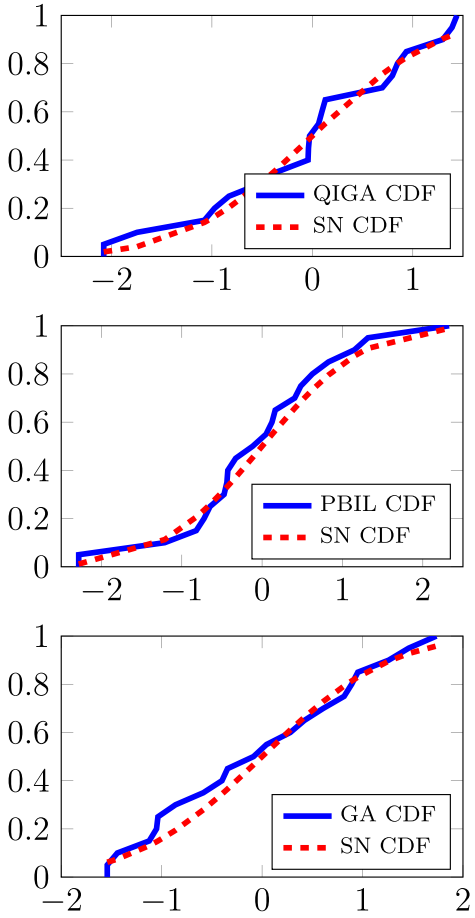


Fig. D13. Instance: 1000, coverage: circle.

Appendix E. Bartlett test for homogeneity of variance

The *homogeneity of variances* is the second assumption to fulfil before deciding what kind of statistical analysis test to perform. For this reason, we perform a Bartlett test.

The level of significance used is 5%. In addition, having three algorithms to compare: QIGA, PBIL and GA, the degree of freedom used is 2. It is calculated using the formula $(Z-1)$, where Z is the number of algorithms to compare. On the basis of the used level of significance and the degree of freedom, the Bartlett test is conducted using a critical value equal to 5.991 according to the *Chi-Square* table. The hypotheses of this test are defined as follows:

- H_0: defines the null-hypothesis that the variance is the same for all the algorithms.
- H_1: defines the hypothesis that the variance is not the same for all the algorithms.

As shown in Tables E1–E5, for each one of the algorithms and for each problem instance, the Bartlett test succeeds to reject the null-hypothesis H_0. Except for the cases of Instance 149 (circle coverage) and Instance 749 (directive coverage), the test fails to reject the null-hypothesis.

Having the condition of homogeneity of variances not satisfied, the assumptions of parametric tests such as the *ANOVA* and the *t-test* cannot be fulfilled. For this reason, we decide to use *non-parametric* statistical tests in the rest of the statistical analysis in the main paper.

Mathematically speaking, the Bartlett test is used to find if a given group of data samples have the same variance. This means that the test performs calculation to find if the interval where the data is distributed is the same (i.e. the length of the intervals delimited by the *upper* and *lower quartiles* and the *median* are the same). Figs. E1–E13 show the distribution of the results obtained by the three algorithms QIGA, PBIL and GA for each problem instance.

Table E1
The Bartlett test for instance 149.

Instance	Coverage	Pv	St	Null-Hypothesis
149	Squared	0.011	9.089	**Rejected**
	Circle	0.368	2.002	**Accepted**
	Directive	$3.928\ 10^{-16}$	70.947	**Rejected**

Table E2
The Bartlett test for instance 349.

Instance	Coverage	Pv	St	Null-Hypothesis
349	Squared	$3.800\ 10^{-6}$	24.961	**Rejected**
	Circle	0.007	10.067	**Rejected**
	Directive	0.003	11.520	**Rejected**

Table E3
The Bartlett test for instance 549.

Instance	Coverage	Pv	St	Null-Hypothesis
549	Squared	$1.666\ 10^{-5}$	22.005	**Rejected**
	Circle	$4.756\ 10^{-4}$	15.302	**Rejected**
	Directive	0.255	2.737	**Rejected**

Table E4
The Bartlett test for instance 749.

Instance	Coverage	Pv	St	Null-Hypothesis
749	Squared	$1.297\ 10^{-12}$	54.742	**Rejected**
	Circle	$1.480\ 10^{-4}$	17.636	**Rejected**
	Directive	0.848	0.329	**Accepted**

Table E5
The Bartlett test for instance 1000.

Instance	Coverage	Pv	St	Null-Hypothesis
1000	Circle	$1.415\ 10^{-16}$	72.988	**Rejected**

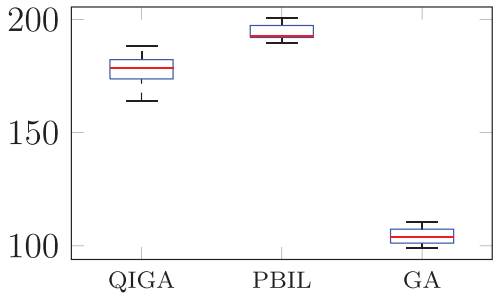


Fig. E1. Instance: 149, coverage: squared.

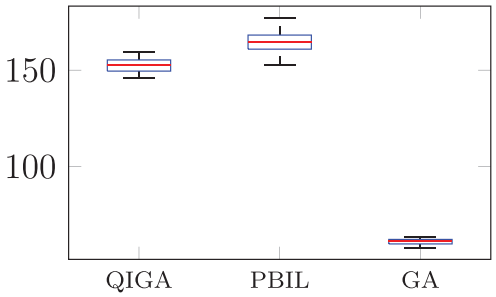


Fig. E2. Instance: 349, coverage: squared.

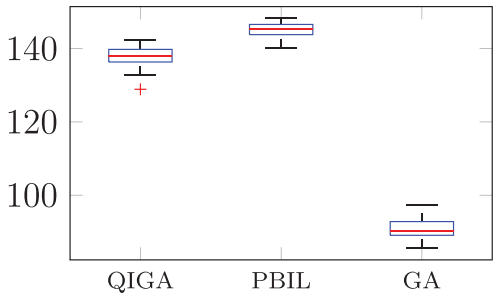


Fig. E3. Instance: 149, coverage: circle.

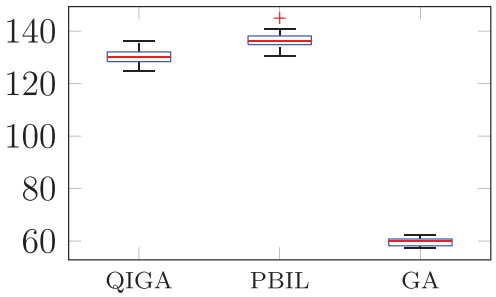


Fig. E4. Instance: 349, coverage: circle.

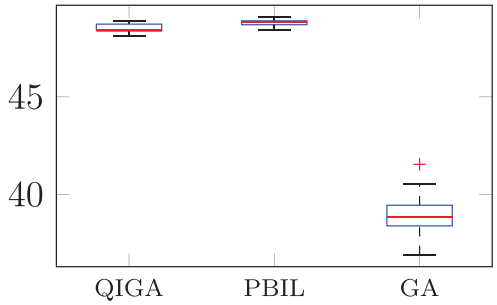


Fig. E5. Instance: 149, coverage: directive.

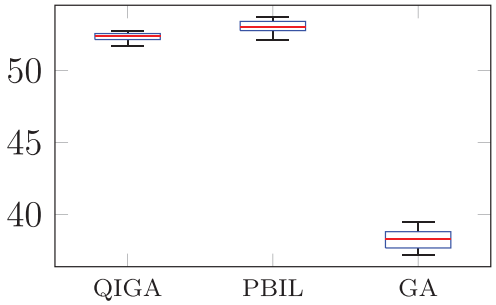


Fig. E6. Instance: 349, coverage: directive.

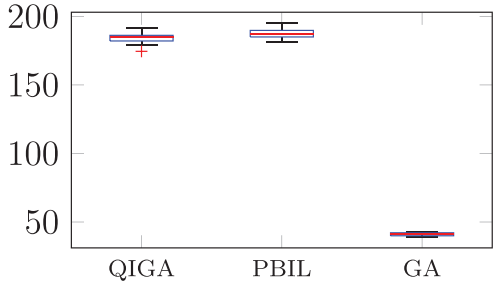


Fig. E7. Instance: 549, coverage: squared.

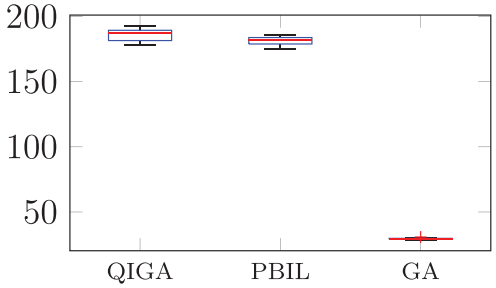


Fig. E8. Instance: 749, coverage: squared.

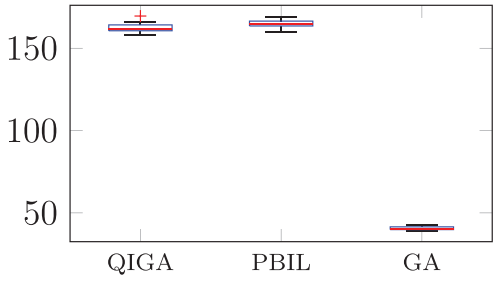


Fig. E9. Instance: 549, coverage: circle.

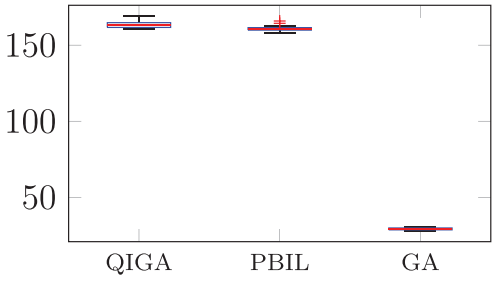


Fig. E10. Instance: 749, coverage: circle.

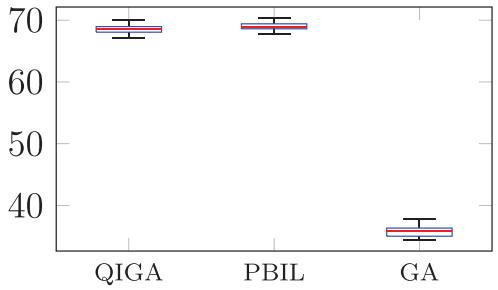


Fig. E11. Instance: 549, coverage: directive.

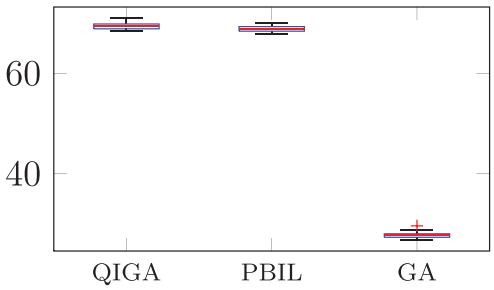


Fig. E12. Instance: 749, coverage: directive.

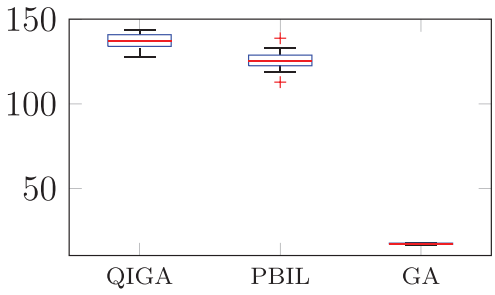


Fig. E13. Instance: 1000, coverage: circle.

Appendix F. Post hoc test

After performing the non-parametric distribution tests (Friedman and Kruskal–Wallis) within our main experimental study in the paper, we found that the distribution of the results obtained by the algorithms are not the same.

So, on the basis of this, we perform a post hoc test to find where the differences occur and ultimately deduce which algorithm is better. The post hoc performs series of pairwise tests between two algorithms to find if there is a difference between their distributions or not. It is based on a statistical ranking of the algorithms according to the results obtained by them in each execution.

Considering that the experiments are performed over 20 executions for each type of coverage, in each instance except Instance 1000, this means that we have 3 types of coverage (i.e 60 executions) except Instance 1000 (i.e. 20 executions). When performing the post hoc procedure, we rank the algorithms according to their performances in each one of the 60 executions. Then, the mean of these ranks is taken. Table F1 shows results of this ranking.

The post hoc is based on the difference between the average rank of the algorithms over the 60 executions in each instance. Figs. F1–F5 show the difference and similarity of the average ranks of the algorithms: QIGA, PBIL and GA for each problem instance.

Table F1
Statistical ranking.

Instance	Algorithm	Mean
149	QIGA	31.45
	PBIL	**49.55**
	GA	10.51
349	QIGA	31.85
	PBIL	**49.15**
	GA	10.51
549	QIGA	35.75
	PBIL	**45.25**
	GA	10.51
749	QIGA	**46.25**
	PBIL	34.75
	GA	10.51
1000	QIGA	**49.20**
	PBIL	31.81
	GA	10.51

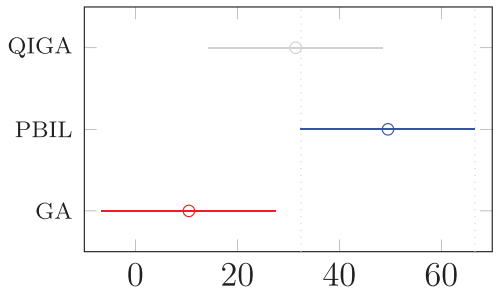


Fig. F1. Instance 149.

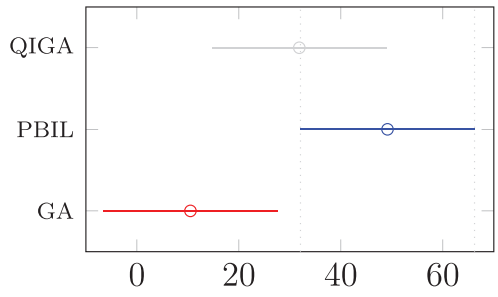


Fig. F2. Instance 349.

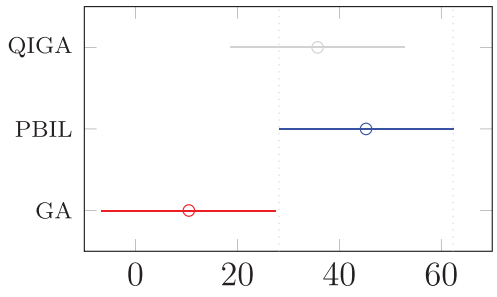


Fig. F3. Instance 549.

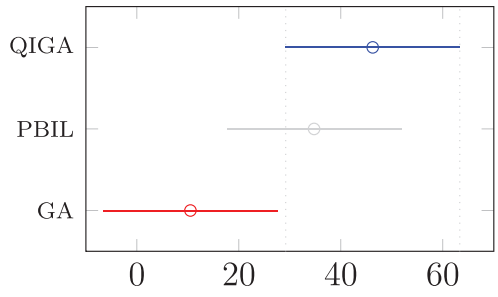


Fig. F4. Instance 749.

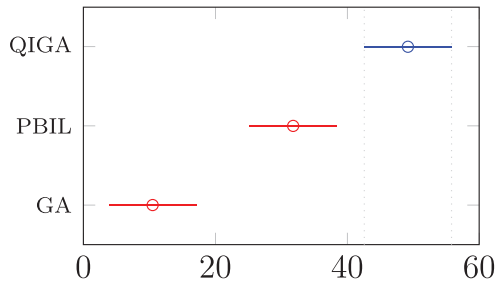


Fig. F5. Instance 1000.

Appendix G. Crossover and mutation probabilities tuning

The crossover P_c and mutation P_m probabilities were tuned through several fine-grained experiments. For scientific relevance, we use the same number of individuals used in the experiments performed within the paper. In addition, each experiment is conducted until reaching the stop criterion of 20,000 fitness evaluations, 100 individuals are used in the population, 200 iterations are performed and repeated for 20 runs.

Our parameters tuning is founded on state-of-the-art studies conducted on GAs. Indeed, these studies show that optimal values of the crossover probability P_c are drawn from the interval [0.5,1]. In addition, these studies show that optimal values of the mutation probability are drawn from the interval $[1/d,0.5]$, where d is the length of the individual in the population. On the basis of that, during our tuning we perform for each instance three tuning steps.

The first step stands in testing three values of crossover probability $P_c \in [0.5,1]$, which are 0.5, 0.6 and 0.7. During this first step both the mutation probability P_m and the learning rate LR of the learning gate have constant values. The latter are set to 0.05 and 0.1, respectively.

The second step stands in using this time three different values of the mutation probability $P_m \in [1/d, 0.5]$, which are 0.04, 0.05 and 0.06. During this second step, both the crossover probability P_c and learning rate LR of the learning gate have constant values. The crossover probability is set to the best value found in the first step. In our case it is 0.7, while the learning rate is set to 0.1.

The third and final step stands in using three different values of the learning rate. These three values are chosen as basic values for the first tuning of the crossover P_c and mutation P_m probabilities. After, a fine-grained sensitivity study is performed to find optimal values of the LR parameter. Thus, 0.1, 0.2 and 0.3 are used in this third step. During this last step, both values of the mutation probability P_m and crossover probability P_c are set to constant values. The latter are the optimal values found in the first and second tuning steps. Thus, P_c is set to 0.7 and P_m is set to 0.05.

On the basis of the "*Mean*" metric of Tables G1–G7, the best results obtained during this tuning are highlighted in bold.

Table G1
Results of Instances 149 and 349 with squared coverage.

Instance 149							Instance 349						
P_c	P_m	*LR*	Best	Worst	Mean	STD	P_c	P_m	*LR*	Best	Worst	Mean	STD
0.5	0.05	0.1	191.353	160.006	**172.035**	7.798	0.5	0.05	0.1	154.455	140.751	147.151	3.674
0.6	0.05	0.1	181.788	156.327	168.866	6.564	0.6	0.05	0.1	151.543	139.906	145.658	3.579
0.7	0.05	0.1	179.496	156.576	168.601	6.333	0.7	0.05	0.1	188.859	175.713	**180.596**	3.929
0.7	0.04	0.1	182.903	159.847	**170.469**	5.939	0.7	0.04	0.1	155.119	140.019	145.768	4.271
0.7	0.05	0.1	179.496	156.576	168.601	6.333	0.7	0.05	0.1	188.859	175.713	**180.596**	3.929
0.7	0.06	0.1	178.934	158.494	170.101	5.241	0.7	0.06	0.1	158.781	140.118	147.717	4.526
0.7	0.05	0.1	179.496	156.576	168.601	6.333	0.7	0.05	0.1	188.859	175.713	**180.596**	3.929
0.7	0.05	0.2	181.325	161.961	172.285	6.088	0.7	0.05	0.2	166.683	140.513	150.299	6.385
0.7	0.05	0.3	184.355	163.806	**173.614**	6.051	0.7	0.05	0.3	157.991	144.892	149.433	3.511

Table G2
Results of Instances 149 and 349 with circle coverage.

Instance 149							Instance 349						
P_c	P_m	*LR*	Best	Worst	Mean	STD	P_c	P_m	*LR*	Best	Worst	Mean	STD
0.5	0.05	0.1	142.221	129.855	135.673	3.396	0.5	0.05	0.1	133.637	120.777	127.211	3.141
0.6	0.05	0.1	138.546	129.157	134.704	2.937	0.6	0.05	0.1	133.497	120.505	127.421	2.903
0.7	0.05	0.1	139.979	131.494	**136.183**	2.775	0.7	0.05	0.1	163.465	153.347	**159.829**	2.761
0.7	0.04	0.1	142.548	130.938	**136.194**	2.834	0.7	0.04	0.1	131.636	121.297	127.191	2.701
0.7	0.05	0.1	139.979	131.494	136.183	2.775	0.7	0.05	0.1	163.465	153.347	**159.829**	2.761
0.7	0.06	0.1	140.883	125.497	133.872	4.496	0.7	0.06	0.1	131.929	122.811	126.993	3.038
0.7	0.05	0.1	139.979	131.494	136.183	2.775	0.7	0.05	0.1	163.465	153.347	**159.829**	2.761
0.7	0.05	0.2	142.903	130.965	136.753	2.986	0.7	0.05	0.2	135.191	124.293	128.862	2.661
0.7	0.05	0.3	140.621	133.674	**137.111**	2.027	0.7	0.05	0.3	133.581	123.992	129.888	2.731

Table G3
Results of Instances 149 and 349 with directive coverage.

Instance 149							Instance 349						
P_c	P_m	*LR*	Best	Worst	Mean	STD	P_c	P_m	*LR*	Best	Worst	Mean	STD
0.5	0.05	0.1	48.519	47.331	47.932	0.324	0.5	0.05	0.1	51.661	50.159	50.996	0.443
0.6	0.05	0.1	48.441	47.394	**48.159**	0.237	0.6	0.05	0.1	52.034	49.942	50.801	0.558
0.7	0.05	0.1	48.438	47.571	48.019	0.266	0.7	0.05	0.1	67.811	64.725	**66.291**	0.853
0.7	0.04	0.1	48.851	47.794	**48.313**	0.255	0.7	0.04	0.1	51.761	50.001	50.932	0.538
0.7	0.05	0.1	48.438	47.571	48.021	0.266	0.7	0.05	0.1	67.811	64.725	**66.291**	0.853
0.7	0.06	0.1	48.296	47.298	47.827	0.298	0.7	0.06	0.1	51.882	50.067	50.945	0.533
0.7	0.05	0.1	48.438	47.571	48.019	0.266	0.7	0.05	0.1	67.811	64.725	**66.291**	0.853
0.7	0.05	0.2	48.811	47.883	**48.241**	0.248	0.7	0.05	0.2	52.279	50.894	51.601	0.356
0.7	0.05	0.3	48.628	47.742	48.224	0.251	0.7	0.05	0.3	52.191	50.508	51.491	0.441

Table G4
Results of Instances 549 and 749 with squared coverage.

Instance 549							Instance 749						
P_c	P_m	*LR*	Best	Worst	Mean	STD	P_c	P_m	*LR*	Best	Worst	Mean	STD
0.5	0.05	0.1	185.649	173.716	179.826	4.232	0.5	0.05	0.1	185.483	174.692	179.095	2.904
0.6	0.05	0.1	186.614	172.786	180.001	4.279	0.6	0.05	0.1	183.782	168.608	178.734	3.343
0.7	0.05	0.1	189.348	171.911	**180.136**	4.916	0.7	0.05	0.1	186.087	173.458	**179.403**	2.938
0.7	0.04	0.1	185.775	175.098	179.801	3.222	0.7	0.04	0.1	185.758	171.004	177.883	3.541
0.7	0.05	0.1	189.348	171.911	180.136	4.916	0.7	0.05	0.1	186.087	173.457	**179.403**	2.938
0.7	0.06	0.1	187.321	172.163	**180.157**	3.565	0.7	0.06	0.1	183.906	168.978	179.152	3.761
0.7	0.05	0.1	189.348	171.911	180.136	4.916	0.7	0.05	0.1	186.087	173.457	179.403	2.938
0.7	0.05	0.2	187.199	177.794	**182.713**	2.314	0.7	0.05	0.2	190.481	174.471	**180.765**	4.559
0.7	0.05	0.3	192.184	175.653	182.351	4.704	0.7	0.05	0.3	184.918	173.407	180.676	3.245

Table G5
Results of Instances 549 and 749 with circle coverage.

Instance 549							Instance 749						
P_c	P_m	LR	Best	Worst	Mean	STD	P_c	P_m	LR	Best	Worst	Mean	STD
0.5	0.05	0.1	164.963	152.726	**160.248**	2.867	0.5	0.05	0.1	169.137	154.552	**159.716**	3.631
0.6	0.05	0.1	163.418	153.768	159.962	2.457	0.6	0.05	0.1	162.906	155.244	158.978	2.078
0.7	0.05	0.1	165.056	154.212	159.997	2.536	0.7	0.05	0.1	165.211	154.687	159.421	2.761
0.7	0.04	0.1	167.537	150.857	159.512	3.589	0.7	0.04	0.1	167.032	153.496	**159.566**	3.606
0.7	0.05	0.1	165.056	154.212	**159.997**	2.537	0.7	0.05	0.1	165.211	154.687	159.421	2.762
0.7	0.06	0.1	165.895	152.661	159.611	3.458	0.7	0.06	0.1	163.224	154.088	158.383	2.213
0.7	0.05	0.1	165.056	154.212	159.997	2.537	0.7	0.05	0.1	165.211	154.687	159.421	2.762
0.7	0.05	0.2	167.272	153.849	**161.718**	3.404	0.7	0.05	0.2	167.051	155.252	160.332	3.156
0.7	0.05	0.3	165.427	155.061	160.709	2.584	0.7	0.05	0.3	165.551	155.756	**160.559**	3.093

Table G6
Results of Instances 549 and 749 with directive coverage.

Instance 549							Instance 749						
P_c	P_m	LR	Best	Worst	Mean	STD	P_c	P_m	LR	Best	Worst	Mean	STD
0.5	0.05	0.1	67.657	64.811	66.261	0.915	0.5	0.05	0.1	68.969	65.683	**67.515**	0.948
0.6	0.05	0.1	67.803	65.389	**66.317**	0.625	0.6	0.05	0.1	68.617	66.065	67.087	0.717
0.7	0.05	0.1	67.088	64.806	66.187	0.616	0.7	0.05	0.1	68.094	64.957	66.811	0.783
0.7	0.04	0.1	68.194	64.756	66.181	0.857	0.7	0.04	0.1	68.192	65.632	67.009	0.745
0.7	0.05	0.1	67.088	64.806	66.187	0.616	0.7	0.05	0.1	68.094	64.957	66.811	0.783
0.7	0.06	0.1	67.903	65.157	**66.444**	0.711	0.7	0.06	0.1	68.806	65.395	**67.589**	0.851
0.7	0.05	0.1	67.088	64.806	66.187	0.616	0.7	0.05	0.1	68.094	64.959	66.811	0.783
0.7	0.05	0.2	68.904	65.779	66.877	0.801	0.7	0.05	0.2	68.861	66.261	**67.563**	0.628
0.7	0.05	0.3	68.317	64.781	**66.901**	0.734	0.7	0.05	0.3	68.575	66.327	67.351	0.732

Table G7
Results of Instance 1000 with circle coverage.

Instance 1000						
P_c	P_m	LR	Best	Worst	Mean	STD
0.5	0.05	0.1	140.856	123.383	131.572	4.861
0.6	0.05	0.1	138.693	121.833	130.998	5.595
0.7	0.05	0.1	141.636	123.508	**132.216**	5.128
0.7	0.04	0.1	141.985	120.688	132.201	5.463
0.7	0.05	0.1	141.636	123.508	132.216	5.128
0.7	0.06	0.1	139.492	123.559	**132.971**	5.113
0.7	0.05	0.1	141.636	123.508	132.216	5.128
0.7	0.05	0.2	138.589	125.904	132.171	4.661
0.7	0.05	0.3	138.353	125.064	**132.706**	4.023

Appendix H. Parameter sensitivity analysis: the learning rate

For further understanding of the influence of the Learning Rate (LR) on the behaviour of the proposed approach, experiments on all instances of the APP are run with several successive values of LR.

Since LR is drawn from the interval [0,1], values 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1 are used. In addition, on the basis of the tuning performed for P_c and P_m, for all the sensitivity analysis experiments, values of crossover and mutation probabilities are set to 0.7 and 0.05, respectively. Besides, for consistency and scientific coherence, we use the same experimental parameters used in our main experimental study. Thus, within the sensitivity analysis experiments, the proposed approach is run until reaching the stop criterion of 100,000 fitness evaluations. The quantum population is composed of 100 quantum individuals and run for 1000 iterations. Finally, each experiment is repeated for 20 executions. Tables H1–H7 display the results of the sensitivity analysis experiments.

Figs. H1–H13 display the fitness evolution of the proposed QIGA when tackling instances of the APP using different values of the learning rate.

Table H1
Results of Instances 149 and 349 with squared coverage.

Instance 149					Instance 349				
LR	**Best**	**Worst**	**Mean**	**STD**	***LR***	**Best**	**Worst**	**Mean**	**STD**
0	170.936	145.936	168.762	6.541	0	113.674	99.674	112.374	3.643
0.1	188.361	163.792	178.058	6.002	0.1	159.492	145.709	152.869	3.897
0.2	187.361	162.792	179.058	6.066	0.2	160.463	145.450	152.869	3.465
0.3	187.016	163.013	178.266	6.057	0.3	161.542	147.426	160.232	3.237
0.4	187.016	162.016	184.266	7.040	0.4	161.426	147.426	160.126	3.570
0.5	182.954	157.954	180.504	6.661	0.5	147.948	133.948	146.598	3.760
0.6	184.722	159.722	182.472	6.382	0.6	154.042	139.042	152.542	4.007
0.7	185.762	162.762	182.800	7.350	0.7	165.359	151.359	164.128	3.422
0.8	191.915	164.021	188.841	8.015	0.8	163.410	149.021	161.543	4.677
0.9	186.881	162.881	181.332	9.002	0.9	157.903	141.267	156.089	4.678
1	194.224	170.456	191.058	7.010	1	146.357	132.357	144.007	4.569

Table H2
Results of Instances 149 and 349 with circle coverage.

Instance 149					Instance 349				
LR	**Best**	**Worst**	**Mean**	**STD**	***LR***	**Best**	**Worst**	**Mean**	**STD**
0	137.554	112.125	135.542	5.987	0	99.317	87.317	98.174	3.135
0.1	142.326	128.911	137.645	3.159	0.1	136.101	124.829	130.299	2.715
0.2	142.461	128.465	138.645	4.545	0.2	130.101	118.829	128.299	4.715
0.3	141.809	129.809	140.566	3.675	0.3	136.225	118.895	129.795	5.133
0.4	143.809	129.809	142.459	3.675	0.4	135.895	123.895	134.795	3.144
0.5	143.640	129.640	142.290	3.675	0.5	136.499	124.499	135.349	3.167
0.6	142.125	128.126	141.025	3.339	0.6	130.335	118.335	129.335	2.938
0.7	147.362	124.362	145.012	6.908	0.7	130.364	118.364	128.995	3.501
0.8	144.756	130.756	142.280	5.105	0.8	132.780	118.232	131.064	4.381
0.9	143.982	129.218	142.039	4.801	0.9	132.233	118.214	130.615	4.136
1	142.568	126.568	139.205	5.692	1	135.319	123.319	132.503	4.783

Table H3
Results of Instances 149 and 349 with directive coverage.

Instance 149					Instance 349				
LR	**Best**	**Worst**	**Mean**	**STD**	***LR***	**Best**	**Worst**	**Mean**	**STD**
0	48.989	48.144	48.224	0.222	0	47.544	46.457	47.439	0.321
0.1	48.867	48.100	48.518	0.209	0.1	52.770	51.740	52.346	0.298
0.2	48.550	48.100	48.350	0.209	0.2	52.549	51.447	52.285	0.485
0.3	48.451	48.165	48.251	0.207	0.3	51.770	51.022	51.346	0.356
0.4	48.646	48.165	48.612	0.107	0.4	52.654	51.237	52.487	0.329
0.5	48.837	48.325	48.770	0.156	0.5	53.245	52.125	53.035	0.298
0.6	48.299	47.379	48.237	0.209	0.6	53.245	52.127	53.037	0.298
0.7	48.658	47.156	48.015	0.257	0.7	53.257	52.238	53.143	0.265
0.8	48.342	47.013	48.271	0.297	0.8	52.543	51.021	52.370	0.459
0.9	48.734	46.850	48.556	0.480	0.9	53.104	52.657	53.077	0.101
1	48.385	48.012	48.344	0.104	1	52.903	50.903	52.710	0.512

Table H4
Results of Instances 549 and 749 with squared coverage.

Instance 549					Instance 749				
LR	**Best**	**Worst**	**Mean**	**STD**	***LR***	**Best**	**Worst**	**Mean**	**STD**
0	79.635	62.635	77.921	4.584	0	53.862	39.862	52.688	3.446
0.1	191.572	174.513	184.026	3.693	0.1	192.802	178.257	186.117	4.554
0.2	190.366	173.986	183.450	3.457	0.2	194.639	181.639	193.289	3.483
0.3	181.659	164.022	179.655	4.213	0.3	194.639	179.988	193.248	3.791
0.4	185.053	168.053	183.203	4.804	0.4	194.639	180.639	193.289	3.675
0.5	181.630	164.630	179.930	4.520	0.5	182.490	168.490	181.240	3.538
0.6	182.328	168.328	180.978	3.588	0.6	183.342	168.342	181.492	4.682
0.7	189.759	172.259	187.924	4.742	0.7	187.023	173.236	185.633	3.681
0.8	188.980	169.021	186.560	6.053	0.8	188.444	173.242	186.873	4.072
0.9	193.647	176.051	191.904	4.731	0.9	181.391	167.322	179.914	3.852
1	189.088	172.088	185.438	6.243	1	175.886	159.886	173.886	4.952

Table H5
Results of Instances 549 and 749 with circle coverage.

Instance 549					Instance 749				
LR	**Best**	**Worst**	**Mean**	**STD**	***LR***	**Best**	**Worst**	**Mean**	**STD**
0	75.457	64.756	74.307	3.031	0	52.818	38.818	51.618	3.443
0.1	169.582	158.267	162.436	2.693	0.1	169.157	160.686	163.650	2.355
0.2	165.094	154.095	163.713	3.138	0.2	165.584	161.691	164.436	0.843
0.3	167.094	154.194	164.094	3.033	0.3	166.013	160.691	164.407	1.055
0.4	165.094	154.014	163.999	2.965	0.4	164.691	161.691	164.391	0.801
0.5	160.621	149.621	159.571	2.874	0.5	167.507	164.507	167.207	0.801
0.6	165.016	151.012	163.514	3.884	0.6	162.073	159.012	161.751	0.827
0.7	168.889	157.079	167.374	3.723	0.7	160.998	157.254	160.573	1.087
0.8	171.654	160.216	170.088	3.515	0.8	160.806	157.056	160.477	0.895
0.9	161.138	150.012	160.207	2.726	0.9	165.581	150.013	162.728	5.583
1	168.381	157.381	167.100	3.088	1	163.482	154.480	162.332	2.852

Table H6
Results of Instances 549 and 749 with directive coverage.

Instance 549					Instance 749				
LR	**Best**	**Worst**	**Mean**	**STD**	***LR***	**Best**	**Worst**	**Mean**	**STD**
0	51.872	48.525	51.534	0.976	0	43.327	40.327	43.027	0.801
0.1	70.023	67.114	68.532	0.688	0.1	71.004	68.455	69.430	0.708
0.2	69.770	66.770	69.320	0.999	0.2	71.137	68.001	70.830	0.826
0.3	69.864	65.450	69.259	1.198	0.3	71.137	68.002	70.830	0.825
0.4	69.770	66.770	69.470	0.801	0.4	71.137	68.137	70.837	0.801
0.5	68.043	65.043	67.743	0.801	0.5	68.608	65.608	68.308	0.801
0.6	67.660	66.467	67.386	0.287	0.6	69.737	65.737	69.283	1.155
0.7	68.145	65.245	67.791	0.893	0.7	70.252	67.655	69.937	0.783
0.8	67.289	62.283	66.821	1.259	0.8	68.844	65.125	68.330	1.265
0.9	67.895	64.642	67.521	0.971	0.9	69.451	65.012	68.878	1.414
1	69.269	66.269	68.969	0.801	1	68.648	65.648	68.348	0.801

Table H7
Results of Instance 1000 with circle coverage.

Instance 1000				
LR	**Best**	**Worst**	**Mean**	**STD**
0	29.178	13.178	27.578	4.297
0.1	143.750	127.559	137.121	4.604
0.2	141.355	125.365	136.315	4.548
0.3	144.445	126.115	136.013	4.215
0.4	141.365	125.365	136.715	4.344
0.5	143.254	127.254	137.754	4.149
0.6	138.302	122.012	136.405	4.772
0.7	138.910	122.515	136.405	5.400
0.8	137.551	121.013	135.923	4.476
0.9	127.832	111.213	126.275	4.192
1	136.500	120.499	134.800	4.402

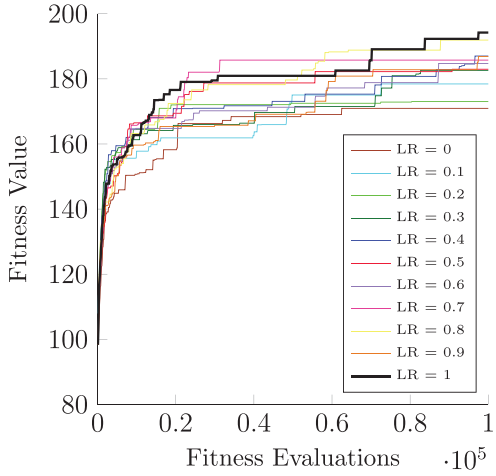


Fig. H1. Instance: 149, coverage: squared.

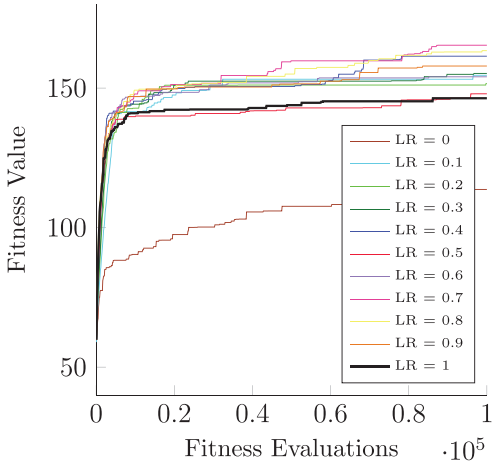


Fig. H2. Instance: 349, coverage: squared.

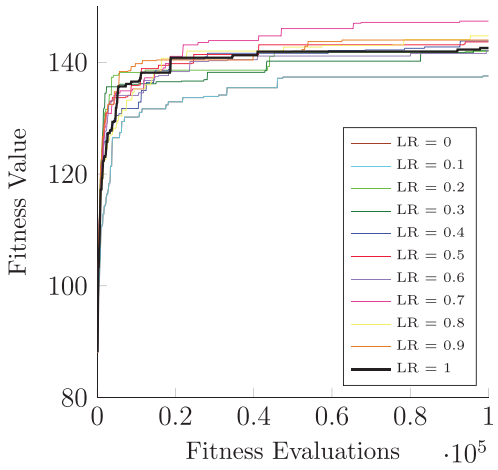


Fig. H3. Instance: 149, coverage: circle.

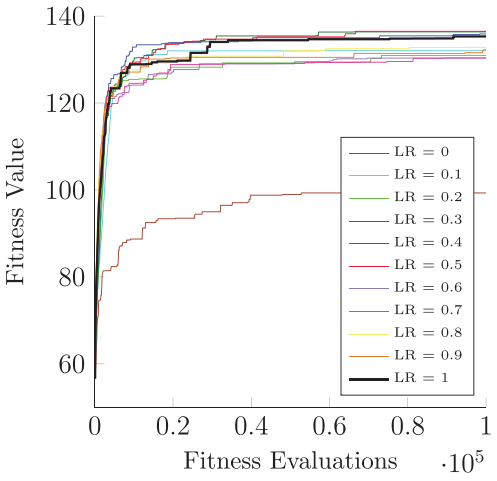


Fig. H4. Instance: 349, coverage: circle.

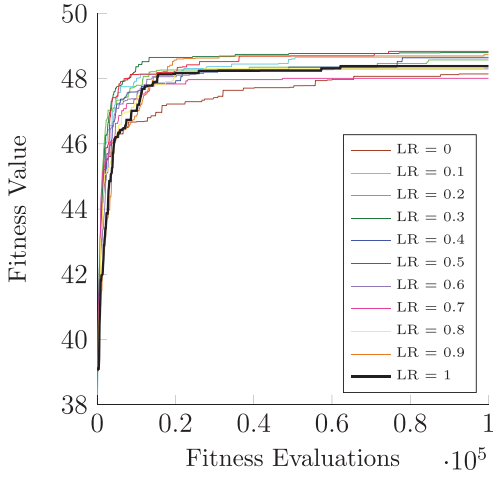


Fig. H5. Instance: 149, coverage: directive.

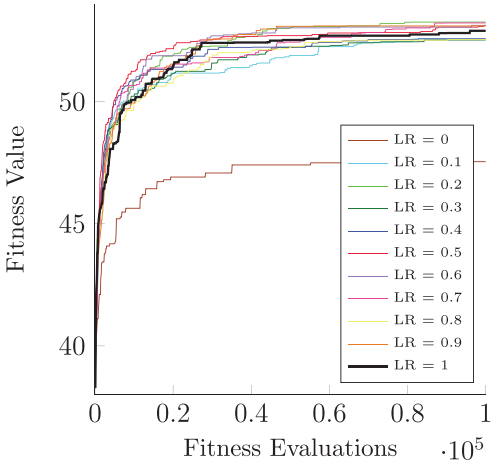


Fig. H6. Instance: 349, coverage: directive.

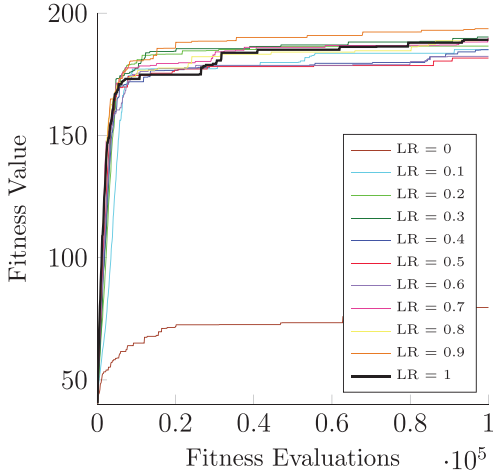


Fig. H7. Instance: 549, coverage: squared.

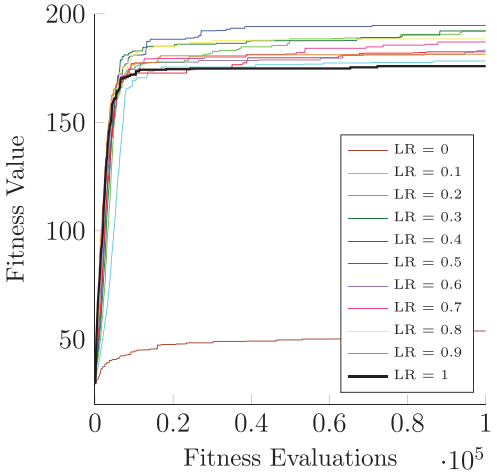


Fig. H8. Instance: 749, coverage: squared.

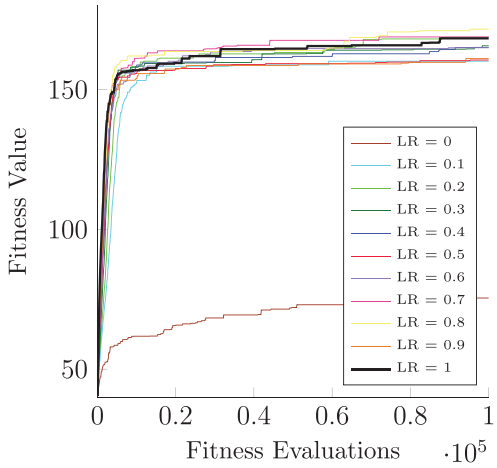


Fig. H9. Instance: 549, coverage: circle.

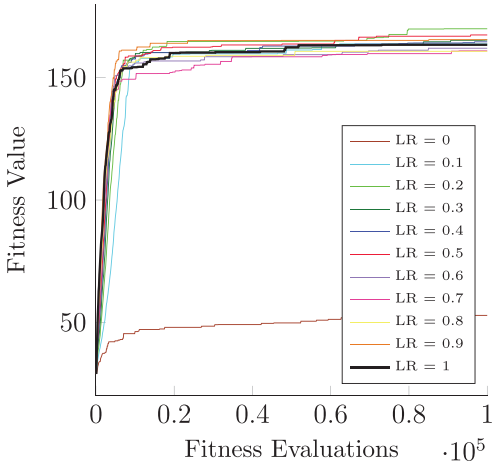


Fig. H10. Instance: 749, coverage: circle.

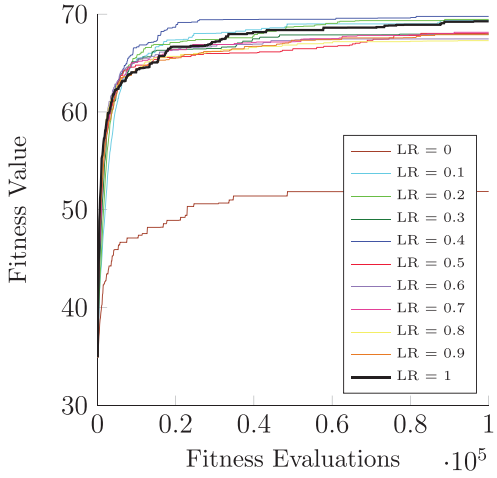


Fig. H11. Instance: 549, coverage: directive.

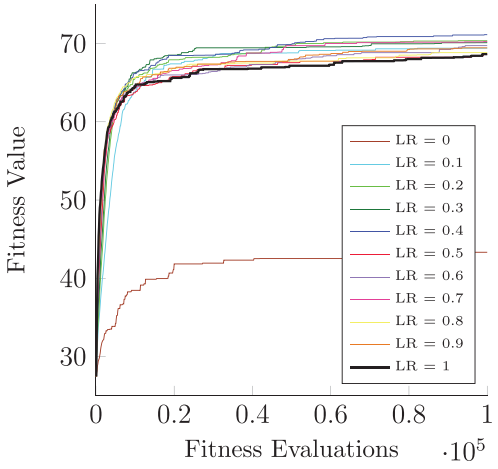


Fig. H12. Instance: 749, coverage: directive.

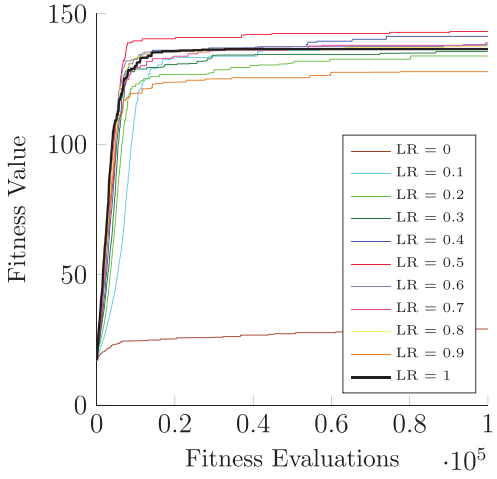


Fig. H13. Instance: 1000, coverage: circle.

References

[1] The Random Benchmark Instances of the STORMS Project Model: Instances of 549 and 749 Positions. ⟨http://www.fichier-rar.fr/2014/10/03/random-instance/⟩.
[2] The Realistic Benchmark Instance of the STORMS Project Model: Instance of 1000 Positions. ⟨http://oplink.lcc.uma.es/problems/rnd.html⟩.
[3] The Synthetic Benchmark Instance of the STORMS Project Model: Instance of 149 Positions. ⟨http://neo.lcc.uma.es/software/more/index.php⟩.
[4] The Synthetic Benchmark Instance of the STORMS Project Model: Instance of 349 Positions. ⟨http://www.httpoplink.lcc.uma.esproblemsrnd.html⟩.

[5] A. Draa, M. Batouche, H. Talbi, A quantum-inspired differential evolution algorithm for rigid image registration, in: Proceedings of the International Conference on Computational Intelligence (ICCI '04), IEEE, 2004, pp. 408–411.
[6] A. Layeb, A hybrid quantum-inspired harmony search algorithm for 0–1 optimization problems, J. Comput. Appl. Math. 253 (2013) 14–25.
[7] A. Malossini, E. Blanzieri, T. Calarco, Quantum genetic optimization, IEEE Trans. Evol. Comput. 12 (2) (2008) 1133–1150.
[8] A. Narayanan, M. Moore, Quantum-inspired genetic algorithms, in: Proceedings of the International Conference on Evolutionary Computation (CEC '96), IEEE, 1995. pp. 61–66.
[9] A.E. Eiben, M. Jelasity, A critical note on experimental research methodology in EC, in: Proceedings of the Congress on Evolutionary Computation (CEC '02), IEEE, 2002, pp. 582–587.
[10] A.J. Nebro, E. Alba, G. Molina, F. Chicano, F. Luna, J.J. Durillo, Optimal antenna placement using a new multi-objective CHC algorithm, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07), ACM, 2007, pp. 876–883.
[11] B. Chamaret, S. Josselin, P. Kuonen, M. Pizarroso, Radio network optimization with maximum independent set search, in: Proceedings of the 47th Vehicular Technology Conference, IEEE, 1997.
[12] B. Rylander, T. Soule, J. Foster, J. Alves-Foss, Quantum Genetic Algorithms, Technical Report, Initiative for Bioinformatics and Evolutionary Studies (IBEST), Department of Computer Science, University of Idaho.
[13] B.B. Li, L. Wang, A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling, IEEE Trans. Syst. Man Cybern. Part B (2007) 576–591.
[14] C. Gu, Q. Tao, A transforming quantum-inspired genetic algorithm for optimization of green agricultural products supply chain network, in: Proceedings of the International Conference on Computer Engineering and Network (CENet '13), vol. 277, Springer, 2014, pp. 145–152.
[15] C. Hui, Z. Jiashu, Z. Chao, Chaos updating rotated gates quantum-inspired genetic algorithm, in: Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS '04), IEEE, 2004, pp. 1108–1112.
[16] C. Segura, E. Segredo, Y. Gonzalez, C. Leon, Multiobjectivisation of the antenna positioning problem, in: Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence (DCAI '11), Springer, 2011, pp. 319–327.
[17] C. Segura, Y. Gonzalez, G. Miranda, C. Leon, A multi-objective evolutionary approach for the antenna positioning problem, in: Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES '10), Springer, 2010, pp. 51–60.
[18] C. Segura, Y. Gonzalez, G. Miranda, C. Leon, Parallel hyperheuristics for the antenna positioning problem, in: Proceedings of the 7th International Symposium on Distributed Computing and Artificial Intelligence (DCAI '10), Springer, 2010, pp. 471–479.
[19] E. Alba, Evolutionary algorithms for optimal placement of antennae in radio network design, in: Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS '04), IEEE, 2004.
[20] E. Alba, F. Chicano, On the behavior of parallel genetic algorithms for optimal placement of antennae in telecommunications, Int. J. Found. Comput. Sci. 16 (2) (2005) 343–359.
[21] E. Alba, G. Molina, J.F. Chicano, Optimal placement of antennae using metaheuristics, in: Proceedings of the 6th International Conference on Numerical Methods and Applications (NMA '06), vol. 4310, Springer, 2006, pp. 214–222.
[22] E. Segredo, C. Segura, C. Leon, On the comparison of parallel island-based models for the multiobjectivised antenna positioning problem, in: Proceedings of the 15th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES '11), Springer, 2011, pp. 32–41.
[23] E. Talbi, Metaheuristics: From Design to Implementation, Wiley, 2009.
[24] E.G. Coffman, Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: a survey, in: Approximation Algorithms for NP-Hard Problems, PWS Publishing Co., 1997, pp. 46–93.
[25] E.G. Rieffel, P. Wolfgang, An introduction to quantum computing for non-physicists, ACM Comput. Surv. 32 (3) (2000) 300–335.
[26] E.L. Lawler, The quadratic assignment problem, Manag. Sci. 9 (4) (1963) 586–599.
[27] G. Chen, H. Jiang, X. Lei, Reconfigurable antenna design optimization based on improved quantum genetic algorithm, in: Proceedings of the XXXIth URSI General Assembly and Scientific Symposium (URSI GASS '14), IEEE, 2014, pp. 1–4.
[28] G. Gutin, A. Punnen, A. Barvinok, Kh.G. Edward, I.S. Anatoliy, The Traveling Salesman Problem and its Variations. Combinatorial Optimization, Kluwer Academic, 2002.
[29] G. Zhang, W. Jin, L. Hu, A novel parallel quantum genetic algorithm, in: Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '03), IEEE, 2003, pp. 693–697.
[30] G.C. Liao, Solve environmental economic dispatch of smart microgrid containing distributed generation system using chaotic quantum genetic algorithm, Int. J. Electr. Power Energy Syst. 43 (1) (2012) 779–787.
[31] H. Liu, H. Motoda, Computational Methods of Feature Selection, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC, Boca Raton, 2008.
[32] H. Talbi, A. Draa, M.C. Batouche, A genetic quantum algorithm for image registration, in: Proceedings of the International Conference on Information and Communication Technologies: From Theory to Applications, IEEE, 2004, pp. 395–396.
[33] H. Talbi, A. Draa, M.C. Batouche, A new quantum-inspired genetic algorithm for solving the travelling salesman problem, in: Proceedings of the International Conference on Industrial Technology (ICIT '04), IEEE, 2004, pp. 1192–1197.
[34] H. Xing, X. Liu, X. Jin, L. Bai, Y. Ji, A multi-granularity evolution based quantum genetic algorithm for QoS multicast routing problem in WDM networks, Comput. Commun. 32 (2) (2009) 386–393.
[35] I. Grigorenko, M.E. Garcia, Calculation of the partition function using quantum genetic algorithms, Physica A: Stat. Mech. Appl. 313 (3–4) (2002) 463–470.
[36] J. Yang, B. Li, Z. Zhuang, Research of quantum genetic algorithm and its application in blind source separation, J. Electron. 20 (1) (2003) 62–68.
[37] J. Yang, B. Li, Z. Zhwang, Multi-universe parallel quantum genetic algorithm its application to blind-source separation, in: Proceedings of the International Conference on Neural Networks and Signal Processing, IEEE, 2003, pp. 393–398.
[38] J. Zhang, H. Li, Z. Tang, Q. Lu, X. Zheng, Z. Zhou, An improved quantum-inspired genetic algorithm for image multilevel thresholding segmentation, Math. Probl. Eng. 2014 (295402) (2014) 62–68.
[39] J. Zhao, H. Liu, X. Chen, T. Chen, A new technology for MIMO detection: the μ quantum genetic sphere decoding algorithm, in: Proceedings of the 10th Annual Conference (ACA '14), IEEE, 2014, pp. 229–241.
[40] J.A.G. Pulido, S. Priem-Mendes, M.A. Vega-Rodriguez, P.J. Cordeiro, J.M. Sanchez-Perez, Processor for measuring radio network design quality, Wirel. Eng. Technol. 2 (3) (2011) 204–211.
[41] J.C. Lee, W.M. Lin, G.C. Liao, T.P. Tsao, Quantum genetic algorithm for dynamic economic dispatch with valve-point effects and including wind power system, Int. J. Electr. Power Energy Syst. 33 (2) (2011) 189–197.
[42] J.H. Holland, Adaptation in Natural & Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, University of Michigan Press, 1975.
[43] K. Liu, Z.Q. Zhu, Quantum genetic algorithm based parameter estimation of PMSM under variable speed control accounting for system identifiability and VSI nonlinearity, IEEE Trans. Ind. Electron. (99) (2014) 1.
[44] K.H. Han, J.H. Kim, Genetic quantum algorithm and its application to combinatorial optimization problem, in: Proceedings of the Congress on Evolutionary Computation, IEEE, 2000, pp. 1354–1360.
[45] K.H. Han, J.H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, IEEE Trans. Evol. Comput. 6 (6) (2002) 580–593.
[46] K.H. Han, J.H. Kim, Quantum-inspired evolutionary algorithms with a new termination criterion H_ϵ gate, and two-phase scheme, IEEE Trans. Evol. Comput. 8 (2) (2004) 156–169.
[47] K.H. Han, K.H. Park, C.H. Lee, J.H. Kim, Parallel quantum-inspired genetic algorithm for combinatorial optimization problem, in: Proceedings of the Congress on Evolutionary Computation, IEEE, 2001, pp. 1422–1429.
[48] L. Spector, H. Barnum, H.J. Bernstein, N. Swamy, Finding a better-than-classical quantum and/or algorithm using genetic programming, in: Proceedings of the Congress on Evolutionary Computation (CEC '99), IEEE, 1999.
[49] L. Zhiyong, L. Zhe, G. Rudolph, On the convergence properties of quantum-inspired multi-objective evolutionary algorithms, in: Proceedings of the 3rd International Conference on Intelligent Computing (ICIC '07), vol. 2, Springer, 2007, pp. 245–255.
[50] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96), ACM, 1996, pp. 212–219.
[51] M. Chiang, Nonconvex optimization for communication networks, in: Advances in Applied Mathematics and Global Optimization, vol. 17, Springer, 2009, pp. 137–196.
[52] M. Mohammadi, F. Nassiri, A. Malek, Optimization Models for Solving Lot-Sizing Problems: A Study in Mixed Integer Programming with Feasible Solutions, LAP LAMBERT Academic Publishing, 2012.
[53] M. Rahoual, P. Siarry, Reseaux informatiques: conception et optimization, Editions Technip, 2006.

[54] M. Sardana, R.K. Agrawal, B. Kaur, Clustering in conjunction with quantum genetic algorithm for relevant genes selection for cancer microarray data, in: Proceedings of PAKDD International Workshops: DMApps, DANTH, QIMIE, BDM, CDA, CloudSD on Trends and Applications in Knowledge Discovery and Data Mining, vol. 7867, Springer, 2013, pp. 428–439.
[55] M. Vasquez, J.K. Hao, A heuristic approach for antenna positioning in cellular networks, J. Heuristics 7 (5) (2001) 443–472.
[56] M. Zhang, Y. Deng, D. Chang, A novel quantum genetic clustering algorithm for data segmentation, in: Proceedings of the Companion on Genetic and Evolutionary Computation Companion Conference (GECCO Comp '14), ACM, 2014, pp. 1485–1486.
[57] M.A. Vega-Rodriguez, J.A.G. Pulido, E. Alba, D. Vega-Perez, S. Priem-Mendes, G. Molina, Evaluation of different metaheuristics solving the RND problem, in: Proceedings of the EvoWorkshops on EvoCoMnet, EvoFIN, EvoIASP, EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog: Applications of Evolutionary Computing, Springer, 2007, pp. 101–110.
[58] M.A. Vega-Rodriguez, J.A.G. Pulido, E. Alba, D. Vega-Perez, S. Priem-Mendes, G. Molina, Using omnidirectional BTS and different evolutionary approaches to solve the RND problem, in: Proceedings of the 11th International Conference on Computer Aided Systems Theory (Eurocast '07), Springer, 2007, pp. 853–860.
[59] M.G.C. Resende, P. Pardalos, Handbook of Optimization in Telecommunications, Springer, 2008.
[60] P. Calegari, F. Guidec, P. Kuonen, A parallel genetic approach to transceiver placement optimisation, in: Proceedings of the Parallel and Distributed Systems (SIPAR Workshop '96), 1996, pp. 21–24.
[61] P. Calegari, F. Guidec, P. Kuonen, Urban radio network planning for mobile phones, EPFL Supercomput. Rev. (9) (1997) 4–10.
[62] P. Calegari, F. Guidec, P. Kuonen, D. Kobler, Parallel island-based genetic algorithm for radio network design, J. Parallel Distrib. Comput. 47 (1) (1997) 86–90.
[63] P. Calegari, F. Guidec, P. Kuonen, D. Wagner, Genetic approach to radio network optimization for mobile systems, in: Proceedings of the 47th Vehicular Technology Conference, IEEE, 1997, pp. 755–759.
[64] P. Calegari, F. Guidec, P. Kuonen, F. Nielsen, Combinatorial optimization algorithms for radio network planning, Theor. Comput. Sci. 263 (1–2) (2001) 235–245.
[65] P. Calegari, P. Kuonen, F. Guidec, Radio Network Planning with Combinatorial Optimisation Algorithms.
[66] P. Dirac, The Principles of Quantum Mechanic, 4th edition, Oxford University Press, 1958.
[67] P. Kuonen, F. Guidec, P. Calegari, A novel parallel quantum genetic algorithm, in: Proceedings of the International Conference on High-Performance Computing (HPC '98), 1998, pp. 277–282.
[68] P.R. Calegari, Parallelization of population-based evolutionary algorithms for combinatorial optimization problems (Ph.D. thesis), Claude Bernard University, 1999.
[69] P.W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in: Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS '94), IEEE, 1994, pp. 124–134.
[70] R. Nowotniak, J. Kucharski, Higher-order quantum-inspired genetic algorithms, in: Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '14), IEEE, 2014, pp. 465–470.
[71] S. Baluja, Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Technical Report, 1994.
[72] S. Dey, S. Bhattacharyya, U. Maulik, Quantum-inspired genetic algorithm and particle swarm optimization using chaotic map model based interference for gray level image thresholding, Swarm Evol. Comput. 15 (2014) 38–57.
[73] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Inc., 1990.
[74] S. Meshoul, K. Mahdi, M.C. Batouche, A quantum-inspired evolutionary framework for multi-objective optimization, in: Proceedings of the 12th Portuguese Conference on Progress in Artificial Intelligence (EPIA '05), Springer, 2005, pp. 190–201.
[75] S. Prakash, D. Vidyarthi, A novel scheduling model for computational grid using quantum genetic algorithm, J. Supercomput. 65 (2) (2013) 742–770.
[76] S. Priem-Mendes, G. Molina, M.A. Vega-Rodriguez, J.A.G. Pulido, Y. Saez, G. Miranda, C. Segura, E. Alba, P. Isasi, C. Leon, J.M. Sanchez-Perez, Benchmarking a wide spectrum of metaheuristic techniques for the radio network design problem, IEEE Trans. Evol. Comput. 13 (5) (2009) 1133–1150.
[77] S. Priem-Mendes, J.A. Gomez-Pulido, M.A. Vega-Rodriguez, A.M. Pereira, J.M.S. Perez, Fast wide area network design optimisation using differential evolution, in: Proceedings of the International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP '07), IEEE, 2007, pp. 3–10.
[78] S. Priem-Mendes, J.A. Gomez-Pulido, M.A. Vega-Rodriguez, J.M. Sanchez-Perez, Y. Saez, P. Isasi, The Radio Network Design Optimization Problem Benchmarking and State-of-the-Art Solvers, Studies in Computational Intelligence, vol. 210, Springer, 2009.
[79] S. Priem-Mendes, J.A.G. Pulido, M.A. Vega-Rodriguez, S.M.D. Jaraiz, J.M. Sanchez-Perez, A differential evolution based algorithm to optimize the radio network design problem, in: Proceedings of the 2nd International Conference on e-Science and Grid Computing (e-Science '06), IEEE, 2006, pp. 119.
[80] S. Yang, M. Wang, L. Jiao, A genetic algorithm based on quantum chromosome, in: Proceedings of the 7th International Conference on Signal Processing (ICSP '04), IEEE, 2004, pp. 1622–1625.
[81] S.N. Sivanandam, S.N. Deepa, Introduction to Genetic Algorithms, 1st edition, Springer, 2007.
[82] S.V. Ulyanov, Quantum soft computing in control process design: quantum genetic algorithms and quantum neural network approaches, in: Proceedings of the World Automation Congress, IEEE, 2004, pp. 99–104.
[83] T. Hey, Quantum computing: an introduction. Comput. Control Eng. J. 10 (3) (1999) 105–112.
[84] U.K. Chakraborty, Computational Intelligence in Flow Shop and Job Shop Scheduling, 1st edition, Springer, 2009.
[85] X.K. Wei, W. Shao, C. Zhang, J.L. Li, Improved self-adaptive genetic algorithm with quantum scheme for electromagnetic optimisation, IET Microw. Antennas Propag. (2014) 8.
[86] Y. Saez, F. Zazo, P. Isasi, A study of the effects of clustering and local search on radio network design: evolutionary computation approaches, in: Proceedings of the 8th International Conference on Hybrid Intelligent Systems (HIS '08), IEEE, 2008, pp. 951–954.
[87] Y. Xu, X. Mei, Z. Dai, Q. Su, Application of the improved quantum genetic algorithm, in: Proceedings of the International Conference of Life System Modeling and Simulation (LSMS '14) and International Conference on Intelligent Computing for Sustainable Energy and Environment (ICSEE '14), Springer, 2014, pp. 122–128.
[88] Z.E.M. Dahi, C. Mezioud, A. Draa, Binary bat algorithm: on the efficiency of mapping functions when handling binary problems using continuous-variable-based metaheuristics, in: Proceedings of the 5th IFIP TC International Conference on Computer Science and Its Applications (CIIA), Springer, 2015, pp. 3–14.