Future Generation Computer Systems 130 (2022) 164–180

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Metaheuristics on quantum computers: Inspiration, simulation and real execution

Zakaria Abdelmoiz Dahi [a,b,*], Enrique Alba [c]

[a] Dep. of Lenguajes y Ciencias de la Computación, Fac. ETSI Informática, University of Malaga, Spain
[b] Dep. of Fundamental Computer Science and its Applications, Fac. NTIC, University of Constantine 2, Algeria
[c] ITIS Software, Edificio Ada Byron, University of Malaga, Spain

ARTICLE INFO

Article history:
Received 31 March 2021
Received in revised form 15 December 2021
Accepted 19 December 2021
Available online 23 December 2021

Keywords:
Discrete variable gate-model quantum computing
Evolutionary algorithms
Hybrid computation paradigms

ABSTRACT

Quantum-inspired metaheuristics are solvers that incorporate principles inspired from quantum mechanics into classical-approximate algorithms using non-quantum machines. Due to the uniqueness of quantum principles, the inspiration of quantum phenomena and the way it is done in fundamentally different non-quantum systems rather than real or simulated quantum computers raise important questions about these algorithms' design and the reproducibility of their results in real or simulated quantum devices. Thus, this work's contribution stands in a first step towards answering those questions as an attempt to identify key findings in the existing literature that should be considered or adapted in order to build hybrid or fully-quantum algorithms that can be used in quantum machines. This is done by proposing and studying four inspired, simulated and real quantum cellular genetic algorithms that, as far as the authors' knowledge, are the first quantum structured metaheuristics studied in the three quantum realms using a quantum simulator with 32 quantum bits and a real quantum machine employing 15 superconducting quantum bits. The users' mobility management in cellular networks is taken as a validation problem using 13 real-world instances. The comparisons have been made against 6 diverse algorithms using 9 comparison metrics. Thorough statistical tests and parameters' sensitivity analysis have been also conducted. The experiments allowed answering several questions, including how quantum hardware influences the studied-algorithms' search process. They also enabled opening new perspectives in quantum metaheuristics' design.

1. Introduction

Quantum computing is a promising field that is shaping the world's future. It was born from incorporating unique mechanics of quantum physics (e.g. superposition, entanglement, etc.) into classical computing [1]. Several tools came out to light from this fusion where quantum algorithms are one of the most efficient ones. These techniques offer a plethora of applications in engineering, economy, health and especially optimisation problem-solving. The latter is one of today's most challenging and omnipresent issues where approximate techniques such as metaheuristics appear also to be a promising alternative that provides a good balance between the solving effectiveness and a quick hands-on that does not require advanced knowledge of the problem [2].

As an attempt to merge, using classical machines, the strengths of both counter-intuitive quantum phenomena and standard metaheuristics, the existing literature proposes quantum-*inspired* metaheuristics (e.g. [3]). However, the peculiar quantum principles that are being inspired and the classical machines where they are being implemented are fundamentally different and therefore, quantum computer simulators and real quantum computers should be used instead. Some reviews and analyses like the one in [4] have already been done on quantum-inspired algorithms, but no proper use of real quantum machines or simulators such as the International Business Machines corporation Quantum experience (IBMQ) has been made yet. Considering these facts, important questions come to light regarding the quantum-inspired metaheuristics' findings, design, results' reproducibility, their possible use within real quantum systems, etc. Among many of these questions, the following can be cited:

- Q1: *Are quantum principles incorporated in quantum-inspired metaheuristics in such a way that their findings still be reproducible if run on (I) quantum computer simulators or (II) real quantum machines*?

* Corresponding author at: Dep. of Lenguajes y Ciencias de la Computación, Fac. ETSI Informática, University of Malaga, Spain and Dep. of Fundamental Computer Science and its Applications, Fac. NTIC, University of Constantine 2, Algeria.
E-mail addresses: zakaria.dahi@uma.es, zakaria.dahi@univ-constantine2.dz (Z.A. Dahi), eat@lcc.uma.es (E. Alba).

https://doi.org/10.1016/j.future.2021.12.015

- Q2: *Since quantum systems provide a theoretical computational speed-up, (I) can the quantum-inspired metaheuristics, with their present design, take advantage of it? (II) If no, How should they be redesigned to do so?*

Considering the recent works done on quantum supremacy [5–7], it is paramount to build relevant and useable techniques that unleash the full potential of quantum systems (e.g. rethinking software engineering for quantum systems, designing efficient quantum algorithms, etc.). In this same line of thoughts, answering the previously-cited questions will help identify the key findings in quantum-inspired metaheuristics' literature that should be considered or adapted so as to design hybrid or fully-quantum solvers useable on quantum machines and able to take profit from the computational acceleration they provide.

The work being presented in this paper attempts to answer the aforementioned questions and do this by devising and studying several *inspired*, *simulated* and *real* Quantum Cellular Genetic Algorithms (QCGA). To the best of the authors' knowledge, this work presents the *first* comparison of this kind and also the *first* structured quantum metaheuristics implemented on a quantum computer simulator with 32 *qu*antum *bits* (qubits) and also on a 15-qubits real quantum machine. The proposal has been assessed for solving the challenging NP-hard mobility management in Global System for Mobile communications (GSM) - Long Term Evolution (LTE) cellular networks. The QCGA's efficiency and scalability have been assessed by performing the experiments over *thirteen* diverse and differently-sized realistic instances. The comparisons have been made against *six* well-known and various solvers. It is believed that several contributions are presented in this work. They can be summarised in two main families:

- C1: As far as the authors' knowledge, this work (I) is the first to propose, study and analyse *qualitatively* and *quantitatively*, within the same work, a metaheuristic in the three realms together: *inspired quantum computing*, *quantum simulators* and *real quantum computers*. (II) Is the first to propose, run and analyse an advanced/structured metaheuristic not only on both an IBMQ quantum computer simulator and a real quantum machine but even in *real* discrete variable gate-model quantum machines in general. (III) Is the first to run an advanced/structured metaheuristic by exploiting and using all the available 32 qubits of the IBMQ quantum simulator and all the 15 qubits of the real IBMQ quantum machine. (IV) Presents proposals that are applicable on *any* discrete variable gate-based quantum computer simulators or real quantum machines.
- C2: In this work (I) an NP-hard real-life problem is tackled, which is the users' mobility management in advanced cellular networks. (II) The experiments are conducted on 13 realistic networks and a new instance inspired from 6 months of real and diverse communication trace is also proposed. (III) The comparisons are made against 6 solvers of different types and complexities (e.g. exact, population-based, etc.), where 9 comparison metrics are considered. (IV) To provide further understanding of the proposals, an in-depth statistical comparison was performed using null-hypothesis tests and a parameter sensitivity analysis that analyses the influence of 3 main parameters on the proposed approaches' behaviour and efficiency.

The remainder of this paper is structured as follows. Section 2 presents basic concepts related to optimisation problems and metaheuristics, cGAs and quantum computing as well as a bird's eye view on metaheuristics in discrete variable gate-model quantum machines. After that, the proposed approach is introduced in Section 3, while Section 4 presents the experimental study in order to assess the performances of the devised QCGAs. Finally, Section 5 concludes the paper.

2. Fundamental background and related works

This section explains the basic concepts related to this present work, including optimisation problems and metaheuristics. Also, it introduces background theory regarding cellular genetic algorithms, quantum computing and a literature review.

2.1. Optimisation problems and metaheuristics

In the following sections, some fundamental concepts are provided for both optimisation problems and metaheuristics.

2.1.1. Problems and neighbourhoods

An unconstrained optimisation problem is a function f that assigns to each solution, s, from the problem's *search space*, $\mathbb{S}$, a quality $f(s)$ called *fitness value*. The aim is to find a *global optimum* solution s_* that satisfies: $\forall\ s \in \mathbb{S}, f(s_*) < f(s)$ (*if minimising*) or $f(s_*) > f(s)$ (*if maximising*). Among many dissimilarities, the main one that exists between continuous ($s \in \mathbb{R}$) and binary problems ($s \in \{0, 1\}$) is the neighbourhood's, $\mathbb{N}_s$, structure of each solution s. $\mathbb{N}_s$ is a set of solutions s' generated by performing a perturbation on s using some search operator(s). In continuous problems, $\mathbb{N}_s$ can be defined as a sphere with a radius $r > 0$. Fig. 1(a) illustrates an example of 3-dimensional continuous problem, where x, y and z are the components of a given solution. The distance between each neighbour $s' \in \mathbb{N}_s$ and the candidate solution s is described in terms of an Euclidean distance expressed by Eq. (1), where $\mathbb{N}_s = \{s' \in \mathbb{R}^3 / \|s' - s\| < r\}$. On the other hand, a 0–1 problem neighbourhood is defined by $\mathbb{N}_s = \{s' \in \{0, 1\} / d(s' - s) \leq r\}$, where d is the Hamming distance between s' and s (It is equal to 1: represents one bit difference). Fig. 1(b) illustrates the case of a 4-dimensional binary problem [8].

$$\|s' - s\| = \sqrt{(x' - x)^2 + (y' - y)^2 + (z' - z)^2} \tag{1}$$

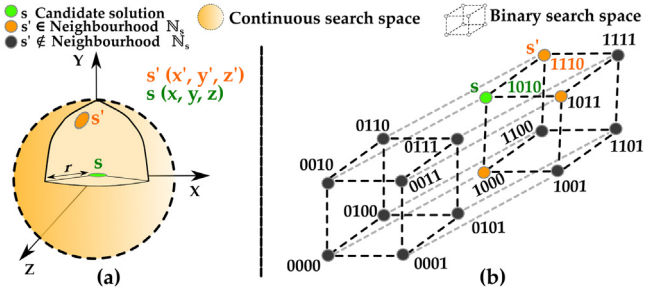

Fig. 1. (a) 3-dimensional continuous and (b) 4-dimensional binary neighbourhoods.

2.1.2. Metaheuristics

Metaheuristics are frequently problem-independent solving methods that start generally by initialising some problem's solution(s) which will be iteratively evolved using search operators to hopefully produce better solution(s) [9]. The metaheuristic's efficiency often depends on how good each of its components suits the problem's neighbourhood features [8]. Thus, the main differences (among possible others) between continuous and binary metaheuristics can be brought to two components: the *solution representation* and the *search operators*. In the following, an analysis is provided about the main approaches proposed to suit binary neighbourhoods at these two levels.

(a) Solution Encoding

Besides algorithms that employ a basic binary representation (see Fig. 2(a)), some mapping techniques, whose the efficiency has been discussed in [10], were also designed to allow continuous metaheuristics to tackle binary problems without changing their solution representation. As a matter of fact, to guide the search, metaheuristics rely on the neighbourhood's structure (e.g. locality [8]) and the objective function (e.g. gradient).

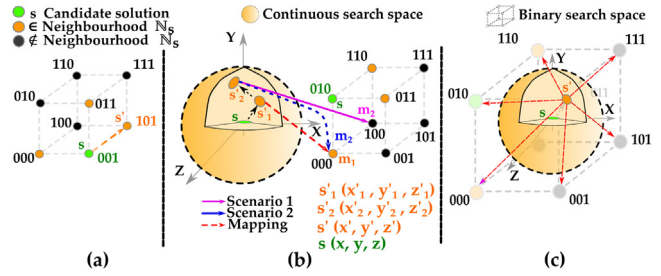

Fig. 2. Solution encoding of a 3-dimensional problem in: (a) binary, (b) mapping-based and (c) distribution-based algorithms.

The first issue with mapping-based metaheuristics is that these two elements are dissociated considering that the search process is made with regard to the continuous domain, while the solution's evaluation and mapping are done with respect to the binary one. Secondly, suppose two continuous solutions s'_1 and s'_2 within the same neighbourhood $\mathbb{N}_s$ of s and their mapped binary counterpart m_1 and m_2. Knowing that s'_2 is obtained after perturbing s'_1, there is no guarantee that m_1 and m_2 will belong to the same neighbourhood. So, necessary concepts like *exploration* and *exploitation* are tricky in such algorithms. Actually, as a hypothesis, it can be stated that the algorithm's search might look coherent in the continuous domain, but not inside the binary one (see Fig. 2(b), scenario 1). The third issue is that the difference between the quality of continuous solutions is a matter of decimals. So, continuous solvers spend time in achieving slight and precise changes that might be lost once the solution is mapped to a binary-coded one which makes that many continuous solutions can be mapped into a single/same binary one (see Fig. 2(b), scenario 2).

The aforementioned facts do not prevent some algorithms that use continuous representation (e.g. quantum-inspired and distribution-estimation algorithms) to efficiently solve binary problems. Mapping-based metaheuristics encode some real numbers without a specific and standard meaning for the binary search space, while the distribution-based ones encode solutions as probabilities of having ones and zeros. This allows both continuous and binary search spaces to coherently merge, where each continuous solution represents some binary ones (Fig. 2(c)).

(b) Search Operators

The efficiency of a given metaheuristic often depends on how well its search operators suit the problem's neighbourhood structure [8]. Therefore, in continuous metaheuristics, they are designed to create a very precise perturbation in the continuous search space (even in terms of decimals) (see Fig. 3(a)). Although it is hard to mathematically model all variation operators of continuous metaheuristics, generally these operators are arithmetic operations between a given number of solutions (*vectorial operations*). The latter can be roughly described by the oversimplified Eq. (2) that represents a holistic, but not absolute formulation where s' is the new updated solution, $s_i/(i = 1, \ldots, n)$ is a solution to the problem being tackled and n is the number of candidate solutions, and o and r are the indices of some of them. Each solution $s_i = \{x_i, y_i, z_i, \ldots\}$, where x_i, y_i, z_i and so on are the components of the i^{th} solution. $\omega_i \in \mathbb{R}/(i = 1, \ldots, n)$ is a random number drawn according to some distribution. As it can be observed in Eq. (2), including already-explored solutions $\{s_o, s_r, \ldots, s_n\}$ often contributes to the *exploitation* of nearby zones, while the random numbers $\{\omega_1, \ldots, \omega_n\}$ control the range of *exploration* of unknown parts of the search space.

$$s' = s_o\omega_o \pm s_r\omega_r \pm \cdots \pm s_n\omega_n \tag{2}$$

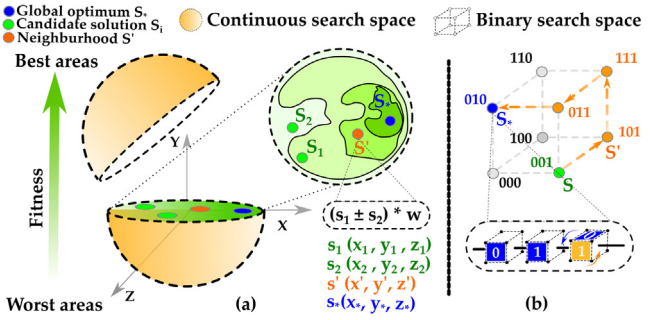

Fig. 3. Search operators in (a) continuous and (b) binary 3-dimensional problem.

Considering now search operators in binary metaheuristics, a simplistic view of a solution to a binary problem might be seen as a key combination of 0s and 1s that the search process aims at breaking down. The most trivial way to do it is to try all the possible combinations in an *exhaustive* (or *methodical*) way (see Fig. 3(b)) which is usually known in the literature as *crossover* and *mutation*. This being said, as seen in the previous section, some mapping-based, quantum-inspired and distribution-based algorithms keep their original search operators in order to evolve binary solutions.

2.2. Cellular genetic algorithms and quantum computing

This section gives some preliminaries about cellular genetic algorithms and quantum computing.

2.2.1. Cellular genetic algorithms

The canonical Genetic Algorithm (GA) is a type of Evolutionary Algorithm (EA) that attempts to reproduce the natural phenomenon of selection and survival of the best beings [9]. It takes an initial population of individuals that represents tentative solutions to the problem being solved, then it evolves them towards fitter states by applying series of operators such as the *selection*, *crossover*, *mutation* and *replacement* part. Many variants of the standard GA have been proposed to enhance its efficiency [11]. Cellular GAs (cGAs) are one of these variants that have been designed to solve complex problems by avoiding local optima. Unlike its classical counterpart, the cGA organises its population in a structured manner (e.g. a 2 or 3-dimensional grid (2D or 3D)), where each node of the grid represents an individual (see Fig. 4) [12].

The cGA starts by *initialising* a population of a given number of individuals. Then, for each individual in the grid, it *selects* some parents' couples in the processed individual's neighbourhood (e.g. Von Neumann, Moore, etc.) in order to go through two successive breeding processes: *crossover* and *mutation*. Both operators are ruled by probabilities P_c and P_m, respectively. The aim of these perturbations is to *exchange*/*alter* the information contained in the selected parents. This gives birth to new offspring that will be *evaluated* using the objective function of the problem

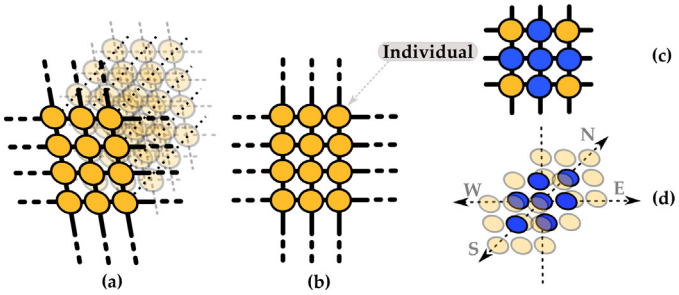

Fig. 4. cGA: (a) 3D and (b) 2D grid, (c) Von Neumann and (d) 3D neighbourhoods.

being addressed. After that, a *replacement* (e.g. synchronous or asynchronous: for all the individuals at a time or one by one in a given order) is done to decide whether they can be considered (or not) to go through the next iteration. This whole process is repeated on each solution in the cGA's grid until the last one which marks the end of one cGA iteration (see Algorithm 1). This procedure is repeated until some termination criterion is met. Besides, it is noteworthy to say that several strategies have been already proposed and can be used at each one of the cGA's steps [13].

Algorithm 1: The cellular genetic algorithm

```
begin
    Initialisation.
    for each iteration do
        for each individual in the grid do
            Selection.
            Variation: crossover and mutation.
            Evaluation.
            Replacement.
        end
    end
end
```

2.2.2. Quantum computing background

Quantum computing was born from the fusion of both computer science and some quantum mechanics' principles (e.g. entanglement, states' superposition, etc.). The latter manipulates qubits which assume that, according to some complementary probabilities α^2 and β^2 that verify Eq. (3), a qubit can be in a quantum state $|\psi>$ that is the superposition of two states simultaneously $|\ 0\ >$ and $|\ 1\ >$ (see Eq. (4)), where α^2 is the qubit's probability to be in the state $|0>$ and β^2 in the state $|1>$.

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

$$|\psi\ >= \alpha|0\ > + \beta|1\ > \tag{4}$$

A wide set of tools came to light from this fusion, and quantum algorithms are one of the most important ones. Taking as an example Shor's factoring [14] and Grovers' search algorithms [1], such techniques turned out to have endless use, especially for overcoming the high-complexity of optimisation problems. Indeed, when using d qubits, features such as the superposition of states allow representing the 2^d binary states simultaneously, which speeds up the computation compared to a representation based on classical bits (see Fig. 5).

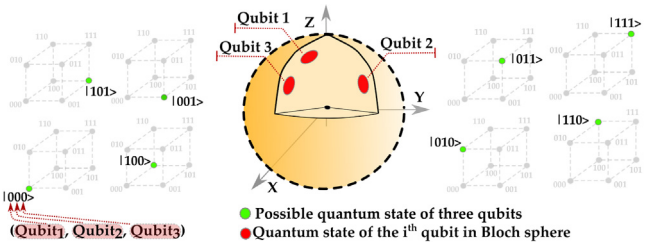

Fig. 5. The superposition of states in quantum computing.

Some decades ago, several technological limitations did not allow implementing "*real*" quantum algorithms. So, researchers such as Narayanan and Moore [15] started by emulating quantum principles within classical techniques, calling them quantum "*inspired*" algorithms. These methods have been investigated in several other works [4]. Later, especially in the last two decades, many quantum computer simulators have been designed such as ProjectQ of Eidgenössische Technische Hochschule (ETH) Zürich, the Quantum Development Kit (QDK) of Microsoft, etc. Also, some real quantum computers were devised such as D-Wave, IBMQ, IonQ, etc. The use of such devices is generally constrained by the limited number of qubits and the restricted access to them (e.g. up to 127 in IBMQ only via IBMQ network) or their high cost (e.g. D-Wave).

In this same line of thoughts, a difference exists between quantum "*inspiration*", "*simulation*" and "*real*" quantum computing. Actually, quantum-inspired algorithms use simple assumptions, probabilities and arithmetic operations as an *attempt* to imitate quantum phenomena on classical machines. On the other hand, *simulation* uses mathematical models to emulate, on classical machines (even simpler quantum machines), some quantum phenomena that would require/happen at low-scale quantum hardware such as transmon superconductors. Finally, *real* quantum computing stands in achieving on real quantum machines what simulation emulates in 0–1 devices. Both real and well-founded simulated quantum computing provide the reliability of their results since everything is done in respect to the laws of physics and mathematics in quantum computing.

Table 1 summarises some differences between inspiration, simulation and real Quantum Computing (QC). It is important to state that since most of the quantum-inspired metaheuristics consider a conceptualisation that is different from the Bloch sphere (see Fig. 6), some side effects might result: (I) the *certainty conservation* law is not properly reflected, (II) some gates are tricky to apply (e.g. the R_z gate defined by Eq. (10)), and (III) the rotation gate R_{insp} used in most quantum-inspired metaheuristics (see Eq. (11)) is different since its counterpart (e.g. R_x, R_y and R_z gates formulated by Eqs. (8)–(10)) consider three distinct plans and are formulated by including imaginary parts in some cases. Indeed, even if the R_{insp} gate is formulated similarly to the R_y one, it is hard to link it to other quantum gates (e.g. R_x and R_z), which make that R_{insp} and R_x, R_y or R_z have a different effect. For instance, suppose a first case where an R_z gate using $\frac{\pi}{4}$ angle is applied on a qubit that is in a superposition state $|\ +\ >$ (see Eq. (14)). This can be done by applying a Hadamard gate H (defined by Eq. (13)) on a qubit in the $|\ 0\ >$ state. Then, apply on that same qubit a T gate defined by Eq. (12) followed by a Hadamard gate H again and a measurement according to the Z axis. The second case tries to recreate the same process as in the first case, but this time in quantum-inspired computing and this by applying the R_{insp} gate with angle $\frac{\pi}{4}$ on a qubit in the

Table 1
Inspired vs. simulated vs. real quantum computing.

Comparison Aspects	Inspiration	Simulation	Real
Decoherence and errors	Not considered	Considered/Not considered	Considered
Qubits' physical connectivity	Not considered	Considered/Not considered	Considered
Entanglement	Not considered	Considered	
Qubit's representation	The values of α and β are real-valued numbers (see Fig. 6(a)).	The values of α and β could be positive, negative or complex numbers (e.g. the states $\|\circlearrowleft>$ and $\|\circlearrowright>$: see Fig. 6(b) and Eqs. (16) and (17)).	
States' superposition	*Assumption* that using a probabilistic representation, all states become superposed.	Reproduced by mathematical models of the real phenomenon.	Phenomenon that happens at the qubit's level.
Axes *X*, *Y* and *Z*	Unclear.	All three axes are considered.	
Interference	The inspired rotation gate R_{insp} has a close formulation to the one of the R_y gate, although technically it is difficult to link it to it or to other gates such as R_x and R_z (see Eqs. (8)–(11)).	Made by applying quantum gates such as R_x, R_y and R_z (see Eqs. (8)–(10)). Technically, it changes the current state $\|\psi>$ by modifying the values of α and β using rotation gates according to the *X*, *Y* or *Z* axes.	

superposition state. Technically, to get a qubit in a superposition state in this second scenario, an R_{insp} gate with angle $\frac{\pi}{4}$ can be applied on a qubit in the $|\ 0>$ state. Then again, an R_{insp} gate with angle $\frac{\pi}{4}$ can be applied on this same qubit. The state vectors that will result in the first and second cases will be different. In the first case (i.e. real QC), a 0.85 probability of getting "0" and 0.15 of getting "1" will be obtained. In the second case (i.e. inspired QC), a 0 probability of getting "0" and a probability of 1 for getting "1" would be obtained. This can be seen by running the first case on a quantum simulator (or real quantum machine) and implementing the second case on a classical machine. Also, it is worth noting that in Eqs. (8)–(11), θ is the rotation angle.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{5}$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{6}$$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{7}$$

$$R_x = \begin{bmatrix} cos(\frac{\theta}{2}) & -i\,sin(\frac{\theta}{2}) \\ -i\,sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix} \tag{8}$$

$$R_y = \begin{bmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix} \tag{9}$$

$$R_z = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{10}$$

$$R_{insp} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{11}$$

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} \tag{12}$$

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{13}$$

$$|+>= \frac{1}{\sqrt{2}}(|0> + |1>) \tag{14}$$

$$|->= \frac{1}{\sqrt{2}}(|0> - |1>) \tag{15}$$

$$|\circlearrowleft>= \frac{1}{\sqrt{2}}(|0> + i\,|1>) \tag{16}$$

$$|\circlearrowright>= \frac{1}{\sqrt{2}}(|0> - i\,|1>) \tag{17}$$

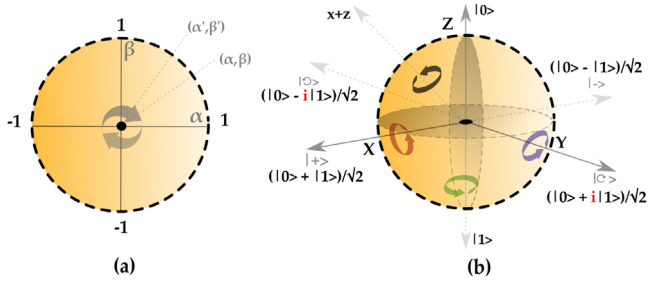

Fig. 6. Quantum states' conceptualisation: (a) inspired and (b) real.

2.3. A bird's eye view on metaheuristics in quantum inspiration, quantum simulators and real quantum machines

The quantum-inspired metaheuristics' design is a research axis that has already a substantial literature and also systematic surveys (e.g. [4]) that anyone can go through, although some works in this axis might suffer from the same shortfalls mentioned in Section 2.2.2. On the other hand, designing and executing quantum metaheuristics on quantum simulators or machines are still in their early stages. Some attempts have been made to devise quantum versions of classical metaheuristics. Such works differ on several aspects such as the type of the quantum simulator/machine being used, the original metaheuristic being studied, the experimental validation conducted, etc. As a first example, the work done in [16] can be cited, where a tentative was made to design a quantum (1+1) EA able to run on a D-Wave quantum machine. The authors of that work have encoded both the (1+1) EA being studied and the problem being solved as an Ising model, that is later solved using the original D-Wave quantum annealer. Technically, this makes the (1+1) EA part of the problem being solved and not the solver. Mathematically speaking, the (1+1) EA becomes as variables within the original formulation of the Quadratic Unconstrained Binary Optimisation problem (QUBO) being tackled. Thus, the proposed quantum (1+1) EA can hardly be linked to the original algorithm or metaheuristics. Besides, few motivation is given to justify the mathematical translation of the

classical (1+1) EA components. Also, no null-hypothesis statistical tests have been conducted to investigate the obtained results. The experiments have showed that the proposed approach attains results that are similar to the ones got by the *classical* (1+1) EA.

This being said, the work conducted in [16] studies the adiabatic QC paradigm which is not the focus of this present work. Indeed, analysing the literature of adiabatic-based quantum metaheuristics goes beyond this paper's scope considering that both discrete variable gate-model quantum simulators and machines are the ones being studied, and more specifically those using superconducting qubits. Regarding the former, the two works that are relevant to this present one are [17,18], where each has presented either a fully or a hybrid quantum Ant Colony Optimisation algorithm (ACO). As validation benchmarks, the authors in [17] tackled the Quadratic Assignment Problem (QAP), while the authors in [18] solved the single-source single-destination shortest-path problem. The authors in [17] have used both simulation and a 2 and 5-qubits real quantum computer. Nonetheless, not much details have been given on the type of simulator used which harden judging the relevance of the simulation done. Also, the QAP being tackled is a QUBO. Several well-known classical and quantum solvers (e.g. D-Wave quantum annealer) already exist and have been specifically designed for solving such problems. In addition, QUBO solvers run on D-Wave quantum systems that allow solving problems requiring up to 5000 qubits, while the size of the QAP instances studied in [17] is only up to 4 qubits on the real quantum machine and up to 6 qubits when using simulation. Also, the devised quantum ACO has been compared only against the classical ACO, and no comparisons against state-of-the-art quantum or classical optimisers have been made. The experiments have been conducted on 5 QAP instances only, and comparisons have been made according to two simple metrics only, while neither basic statistical comparisons (e.g. best, worst, etc.) or null-hypothesis statistical tests have been performed to support the obtained results. The devised quantum ACO has limited applicability since it is strongly constrained by the physical (i.e. hardware) couplings of the qubits on the quantum machine being used. The latter needs to have a certain qubits' configuration to be able to execute the quantum ACO on it. Now, moving to the work done in [18], the authors have used a quantum simulator and a 15-qubits real quantum machine, although it seems that only 4 qubits have been employed in the proposal/study. The experimental validation done suffers also from some of the shortfalls indicated in the case of [17] and this in terms of comparison with state-of-the-art solvers, number of problem instances being studied, the comparison metrics and missing statistical tests. It is worth stating also that in [17,18], no further study is performed to provide an analysis and understanding of the proposed algorithms' behaviour and efficacy.

In order to avoid the shortfalls made in [17,18], as far as the authors' knowledge, this present work is the first in the literature to study and analyse *theoretically* and *numerically* a metaheuristic in the three realms, all together: *inspired quantum computing*, *quantum simulators* and *real quantum computers*. It is also the first to propose, run and analyse a structured metaheuristic not only on both an IBMQ quantum simulator and a real quantum machine, but even in *real* discrete variable gate-model quantum machines in general. The IBM's well-established 32-qubits quantum simulator (`ibmq_qasm_simulator`) and a 15-qubits real quantum machine (`ibmq_16_melbourne`) have been used. The proposals have been run by exploiting and using all the available 32 qubits of the quantum simulator and all the 15 qubits of the real quantum machine. The proposed approaches were designed to be hardware-restriction-free and applicable on *any* discrete variable gate-model quantum simulators or real quantum machines. A challenging NP-hard real-life problem in advanced cellular networks has been tackled. The experiments have been conducted on 12 realistic networks, and a new instance inspired from a 6-months real communication trace was also proposed. The comparisons have been made against 6 solvers of different types, where 9 comparison metrics were considered. To provide further understanding of the proposal, supplementary analyses have been performed by conducting an in-depth statistical comparison using null-hypothesis tests and a parameter sensitivity analysis that investigates the influence of 3 parameters on the proposed approaches' behaviour and efficiency.

3. The proposed quantum cellular genetic algorithms

This work proposes and studies four quantum algorithms. The first two proposals are quantum-*insp*ired: (I) the QCGA that uses the *orig*inal rotation gate often utilised within quantum-inspired metaheuristics (*insp*-QCGA-*orig*), and (II) the QCGA that uses an *inv*erted version of the rotation gate (*insp*-QCGA-*inv*). It is worth noting that these two first techniques were executed on classical machines and will be referred to them as *insp*-QCGA-∗. The third proposed approach is the quantum-*sim*ulated QCGA that is executed on a 32-qubits IBMQ *sim*ulator (*sim*-QCGA). The last devised algorithm is the QCGA that is executed on a 15-qubits IBMQ *real* quantum machine (*real*-QCGA). The Algorithms 2 and 3 illustrate the main framework and differences between the devised approaches. It is to keep in mind that the *insp*-QCGA-∗ algorithms are based on a literature review, including the one given in [4], on how quantum-inspired algorithms are usually devised.

Algorithm 2: *insp*-QCGA-*

```
begin
    Initialisation.
    for each iteration do
        for each individual do
            Selection.
            Interference.
            Crossover.
            Mutation.
            Measurement.
            Evaluation.
            Replacement.
```

Algorithm 3: *sim*-QCGA AND *real*-QCGA

```
begin
    IBMQ-initialisation.
    for each iteration do
        for each individual do
            Selection.
            IBMQ-interference.
            Crossover.
            Mutation.
            IBMQ-measurement.
            Evaluation.
            Replacement.
```

The *insp*-QCGA-* is built by considering the theoretical analysis on the components that could be considered when devising a solver for binary problems (see Section 2.1). This includes (I) considering a probabilistic representation of 0–1 solutions,

(II) structuring the population and (III) using mutation and crossover. Thus, the cGA is the backbone of the *insp*-QCGA-* algorithms being proposed here. All of them share (I) the same concepts (e.g. structured population and interactions limited to a neighbourhood) and (II) search steps. Indeed, the *insp*-QCGA-* evolve a square-torus-like structured population where each grid's node represents an individual (see Fig. 4(b)). The *insp*-QCGA-* phases are performed on each individual by browsing the grid's nodes one by one. Also, the individuals involved in each of these phases are limited to the Von Neumann neighbourhood of the processed node. Furthermore, in addition to the phases existing in a canonical cGA, the *insp*-QCGA-* perform two additional ones based on quantum physics: *interference* and *measurement*.

Regarding both *simulated* and *real* QCGA, as it can be seen from Algorithms 2 and 3, the *insp*-QCGA-*, *sim*-QCGA and the *real*-QCGA share the same search process, although in the *sim*-QCGA and *real*-QCGA, quantum-related phases (coloured in blue) are executed using the 32-qubits IBMQ simulator and a 15-qubits real IBMQ machine, respectively. Thus, in the following, the main components of the *insp*-QCGA-* will be explained first. Later, Section 3.7 explains how the *sim*-QCGA and the *real*-QCGA execute their quantum-related components. Also, it is worth noting that other quantum-related features, such as the *superposition* of states, are considered in this work in order to achieve real simulation of the *sim*-QCGA or real execution in the *real*-QCGA.

3.1. Initialisation and solution encoding

The *insp*-QCGA-* start by initialising a population of n real-coded individuals. Each individual s_i is a possible solution to the problem being addressed, where $s_i = \{q_{i,1}, \ldots, q_{i,d}\}$, $i = \{1, \ldots, n\}$ and d is the problem's size. Unlike the classical binary representation where each element has to be either 1 or 0, the proposed approach uses a probabilistic representation, where each element $q_{ij}/j = \{1, \ldots, d\}$ is a qubit.

Once the initialisation has been performed, the *quantum measurement* is applied in order to get the binary counterpart of the quantum individuals (see Section 3.5). The measured individuals are evaluated using the objective function of the problem being solved. After that, the best individual s_* is extracted. Once this is done, the initial population goes through the search operators until a given stop criterion is met. This whole process is performed sequentially by browsing each node in the *insp*-QCGA-* grid.

3.2. Selection

For each node of the grid, a couple is created. It is composed of the processed individual and another individual that is chosen by performing a binary tournament on the Von Neumann neighbourhood of the processed node (see Fig. 4(c)). This neighbourhood has been chosen instead of another because it is among the most widely-used ones in the literature [19].

3.3. Variation: Crossover and mutation

After the selection, the created couples go through a two-point crossover. This operator gives birth to two new offspring (see Fig. 7(a)), where switch # 1 and 2 represent the indices of the qubits where parts of both parents are swapped. After, both offspring undergo mutation. In this work, a qubit-flip mutation is used. The former flips the probabilities α^2 or β^2 to be in states $|0>$ or $|1>$ (see Fig. 7(b)). Once the mutation is accomplished, two new offspring are produced. Simple crossover and mutation operators have been used on purpose in order to build the most general and understandable algorithm. Also, because a rich literature on both operators exists (e.g. [20]), which facilitates their understanding and control.

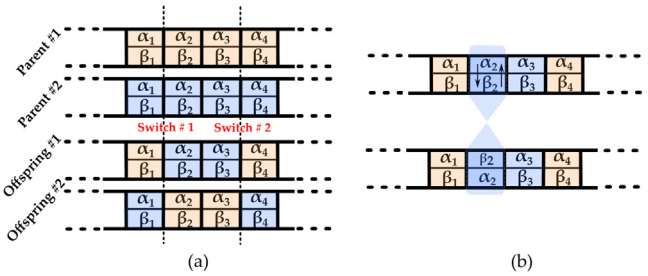

Fig. 7. Variation: (a) crossover and (b) mutation.

3.4. Interference

This phase achieves the quantum phenomenon of *interference*. The aforementioned alters/updates the qubits' values α and β to new values α' and β' in order to make the qubit converge towards either the state $|1>$ or $|0>$. To do so, this work studies the *rotation gate* $\mathtt{R}_{\mathtt{insp}}$ defined by (18) according to an angle θ [21]. Like for the crossover and mutation, this quantum gate has been chosen for the sake of simplicity and for its popularity. As a matter of fact, consistent literature that investigates this gate exists (e.g. [22]), which ease its understanding and control.

$$\begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{18}$$

Although, based on some off-line experiments, it has been decided to use also what is called the "*inverted*" rotation-gate defined by Eq. (19). The latter produces the opposite behaviour of the original rotation gate which turned to be more suitable and allows the *insp*-QCGA-* to be more efficient (see Fig. 8).

$$\begin{bmatrix} \beta' \\ \alpha' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{19}$$

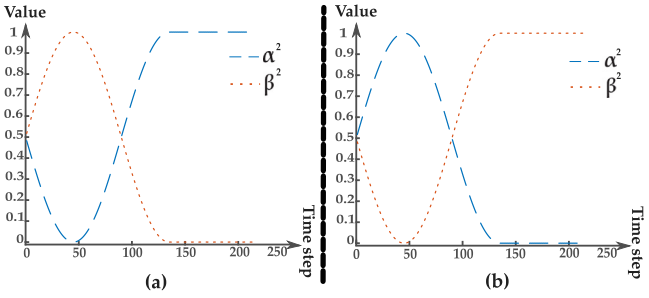

Fig. 8. Evolution of α^2 and β^2 values via Rotation gate: (a) original and (b) inverted.

The use of both versions of the rotation gates results in giving birth to the two proposed variants of the *insp*-QCGA-* called *insp*-QCGA-*orig* and *insp*-QCGA-*inv*. The first one applies the original rotation gate, while the second uses the inverted gate, respectively.

3.5. Measurement

Solutions to binary problems are encoded as 1s and 0s, where each state has a specific significance. On the other hand, the

insp-QCGA-* employ a probabilistic representation. So, a mapping process between continuous and binary encoding is needed in order to switch back to the binary representation. This phase is called *quantum measurement* [22]. In this step, each qubit is set to 1 or 0 according to the condition given below, where ω is a random number drawn from a standard uniform distribution. The measurement done in *sim*-QCGA and *real*-QCGA is explained later in Section 3.7.2.

$$|\psi> \rightarrow \begin{cases} 0, & \text{if } \omega \leqslant \alpha^2 \\ 1, & \text{Otherwise} \end{cases}$$

3.6. Evaluation and replacement

After measuring both offspring (i.e. switching from qubits to bits' representation), an evaluation phase is performed in order to assess their quality. This is done using the objective function of the problem being addressed (see Section 4.1). Now, once the evaluation step is done, the replacement phase takes place in order to decide what will be the composition of the population for the next iterations. For this purpose, the *insp*-QCGA-* algorithms follow the *synchronous* replacement strategy, where the best of the two measured offspring is compared to the individual being processed. Then, the fittest one takes place in an auxiliary population (see Fig. 9).

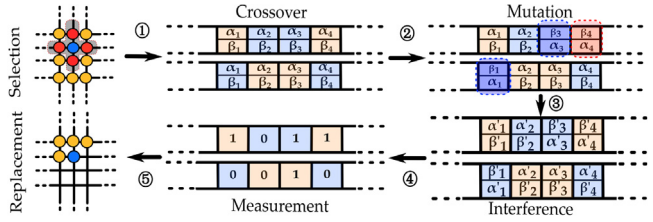

Fig. 9. The *insp*-QCGA-* algorithms' search process.

The whole process of selection, variation, interference, measurement, evaluation and replacement is reiterated until all nodes of the grid are browsed. This marks the end of an *insp*-QCGA-* iteration. After that, the auxiliary population is considered as the new population that goes through the next iteration.

3.7. The superposition and interference in the proposed real and simulated quantum cellular genetic algorithms

This section explains the main motivation behind the use of the IBMQ quantum simulator and real machine rather than other quantum platforms or machines. Also, some details are given on the quantum circuit applied within the *sim*-QCGA and the *real*-QCGA. In this case, it should be kept in mind that the individuals will be encoded using $0 \leqslant \theta \leqslant \pi$, where θ will be in degrees.

3.7.1. The used quantum machine and simulator

IBM is one of the most important manufacturer granting free, but still constrained, access to its quantum simulator and some quantum machines. Thus, in this study, the experiments have been performed on both (I) IBMQ 32-qubits quantum simulator called `ibmq_qasm_simulator` (v0.1.547) and (II) IBMQ 15-qubits real quantum machine called `ibmq_16_melbourne` (v2.1.0) (see Fig. 10).

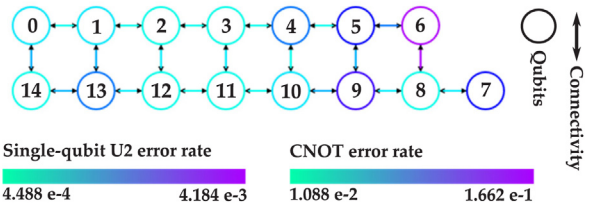

Fig. 10. The features of `ibmq_16_melbourne` machine.

3.7.2. The quantum circuit within the devised real and simulated quantum cellular genetic algorithms

As explained before, the quantum phenomena in *insp*-QCGA-*orig* and *insp*-QCGA-*inv* are emulated, on purpose, like it is done in most of the literature devising quantum-"inspired" meta-heuristics. However, these same quantum events are then (I) simulated within the *sim*-QCGA using the IBMQ 32-qubits simulator and (II) executed in *real*-QCGA using a real IBMQ 15-qubits quantum machine. In both cases, this is done using the following quantum circuit (see Fig. 11):

- **Superposition**: the Hadamard gate (H) is applied.
- **Interference**: a combination of gates is used: Hadamard (H) (see Eq. (13)), the $R_z(\theta)$ (see Eq. (10)) and then Hadamard again.
- **Mutation**: instead of using the Pauli-X (see Eq. (5)) or C-NOT gates, the angles of the rotation gate are inverted to properly reproduce the behaviour of the *insp*-QCGA-* variants.
- **Crossover**: the classical one is kept instead of using a SWAP or Fredkin gate-based implementation. This was done to (I) avoid the circuit's depth and hardware consumption to increase beyond the machine/simulator capacities, and (II) emphasizing our work on extracting the influence of the superposition, measurement and interference.
- **Measurement**: the measurement is applied according to the Z axis.

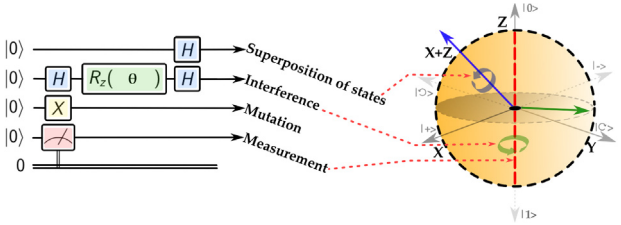

Fig. 11. IBMQ quantum circuit within the *sim*-QCGA and *real*-QCGA.

4. Experimental design, results and analysis

This section presents a brief introduction to the validation problem as well as the experimental design and results used to assess the performances of the proposed algorithms.

4.1. Benchmark problem: The users' mobility management

Cellular networks are one of the most used communication means. The popularity and affordability of their services have made the mobile phone industry a competitive field where service quality is key. Thus, any factor, like bandwidth saturation that jeopardises this quality of service has a tremendous economical relapse. In this same line of thoughts, recent works stated that when trying to locate a mobile user, the generated messages occupy 33% of the bandwidth [19]. The latter is an economically

valuable resource. These facts, among many, made the users' mobility management one of the most challenging problems in cellular networks.

When attempting to find a user to whom a service is meant, the network has to locate it first by sending *polling* messages. In the same way, the mobile user has to inform the network of each of his movements by sending *Location Update* messages (LUs). Therefore, the Mobility Management Problem (MMP) stands in reducing the flow of polling and LU messages. Many schemes for performing polling cycles and location updates exist, but in this work, the most practical one is considered: the *reporting cell scheme* [23]. In this case, when having a set of cells, the MMP consists in labelling each one of them as *reporting* or *non-reporting*. Therefore, the MMP stands in finding the most optimal configuration of reporting and non-reporting cells. This problem is defined by Eq. (20):

$$\min_{s=\{q_1,\ldots,q_d\}} \quad f(s) = \delta * \sum_{i \in B} U_i + \sum_{j \in C} G_j * V_j \tag{20}$$

The fitness function, f, is used to assess the quality of a given solution s, where B is the set of reporting cells and C is the set of all the network's cells. The value of U_i is the location update associated to the i^{th} cell, while G_j and V_j are, respectively, the paging cost and the vicinity of the j^{th} cell. The parameter δ is a weight used to balance between paging and LUs. For a reliable study, the same value employed in the literature (δ = 10) [24] is also used here in this work.

4.2. Experimentation's design

Experiments were run on a cluster of 32 heterogeneous machines in terms of processor (2, 4, 8 and 48 cores) and memory (from 690 to 100049 GB). The implementation was done and run using Matlab 2017 and Python 3.6 under Linux.

Experimentation was performed over 12 networks with increasing size; from 4×4 to 10×10 cells. These networks were previously used in [19] and have been chosen on purpose to assess the proposed algorithms based on several aspects such as the scalability when solving problems of different sizes, complexities, etc. First, in order to demonstrate the motivation behind the use of the inverted rotation gate (see Section 3.4), a comparison is performed between the *insp*-QCGA-*orig* and the *insp*-QCGA-*inv*. All the devised algorithms were then compared against some MMP's solvers which have been selected to represent a wide range of features (see Section 2.1.2). Various examples of these solvers can be cited: distribution-based ones such as the Population-Based Incremental Learning algorithm (PBIL), those using mapping techniques like the Artificial Bee Colony algorithm (ABC), those using a deterministic representation such as the GA [25] and finally exact ones like the CPLEX or other types such as the Greedy Randomised Adaptive Search Procedure (GRASP) or the Nonlinear Optimisation Mesh Adaptive Direct search algorithm (NOMAD) [26].

It is also important to remind that the main objective behind this work is not designing a solver that outperforms the most advanced and specialised algorithms in the literature. So, throughout this work, very trivial settings and components are used at each of the *insp*-QCGA-*, *sim*-QCGA or the *real*-QCGA's steps to show that the proposal does not require advanced parameter settings to display promising efficiency. The settings applied are proved by mathematical demonstration, logical inference or extracted from the literature. This being said, all the devised algorithms evolve a grid of 20×20 nodes, where P_c was set to 1 and P_m to 0.1, while θ was set to 1 based on a mathematical demonstration.

For the sake of a neutral and reliable comparison, the experiments' settings were taken from the works where the MMP's state-of-the-art solvers were proposed. Consequently, the tests were run over 30 executions. In each execution, every approach was run until reaching 175000 fitness evaluations. Several results were reported such as the `best`, `worst`, `mean` and `STandard Deviation` (`STD`) through the executions. Besides that, a series of thorough statistical tests were performed to support the results given according to the aforementioned metrics. The high-probable non-normality of the results' samples is assumed, and thus, the Kruskal–Wallis test is applied as a non-parametric ANOVA test. Considering the results of this former test, a Post-Hoc test (PH) is performed later to know which algorithm is better. Both statistical tests are conducted using a significance level of 5%.

4.3. Numerical results and analysis

This section presents three groups of experiments. The first one in Section 4.3.1 considers the *insp*-QCGA-*orig* and *insp*-QCGA-*inv*. Then, Section 4.3.2 is dedicated to "*simulation*" where experiments are conducted on the best *insp*-QCGA-* algorithms; the *insp*-QCGA-*inv* that simulates in a real manner the quantum phenomena using the IBMQ simulator via Qiskit [27]. Finally, in Section 4.3.3, the *real*-QCGA is executed on an IBMQ real quantum machine. In the two first classes of experiments (Sections 4.3.1 and 4.3.2), the comparison metrics include the `Best`, `Worst`, `Mean`, `STD`, `# Hits`, `Time`, H_0 and `PH`. The first four metrics represent the minimum, maximum, average and the standard deviation of the fitness values obtained throughout the algorithm's iterations. The metric `# Hits` represents how many times an algorithm reaches the best fitness known in the literature of a given instance. The metric `Time` stands for the average time (in seconds) needed for an algorithm to complete one single execution. The metrics H_0 and `PH` have nothing to do with the Hadamard gate (`H`) explained in Section 2.2.2. Indeed, they represent the results of the Kruskal–Wallis ANOVA test and the post-hoc test, respectively. The sign "+" in the column H_0 indicates that the null-hypothesis of the test has been accepted, while the symbol "–" means the opposite. It is to be noted also that, for all the metrics (except H_0), the best results that have been obtained were highlighted in green.

4.3.1. Results of the quantum-inspired cellular genetic algorithms

In these first experiments, a comparison is done between the *insp*-QCGA-*inv*, the *insp*-QCGA-*orig*, PBIL, ABC, GA, GRASP, NOMAD and CPLEX in order to (`I`) compare the proposed quantum-inspired algorithms (*insp*-QCGA-*) against classical ones, and (`II`) extract which one of the devised quantum-inspired algorithms is the best. It is to be kept in mind that the main goal of these first experiments is not only to establish the superiority or inferiority of a given solver regarding the others, but are also meant to answer the research questions established in Section 1.

Most works where the PBIL, ABC, GA, GRASP, NOMAD and CPLEX have been proposed report only the *best* results achieved through 30 executions and no other details of their experiments have been made available. Thus, for the sake of consistency, as a first comparison strategy, the best result is considered as

Table 2
Best fitness attained: *insp*-QCGA-* vs. classical algorithms.

Network	*insp*-QCGA-*orig*	*insp*-QCGA-*inv*	NOMAD	CPLEX	GRASP	PBIL	GA	ABC
4×4 (1)	**98535**	**98535**	**98535**	**98535**	**98535**	**98535**	**98535**	**98535**
4×4 (2)	**97156**	**97156**	**97156**	**97156**	**97156**	97262	**97156**	**97156**
4×4 (3)	**95038**	**95038**	**95038**	**95038**	**95038**	**95038**	**95038**	**95038**
6×6 (1)	177869	**173701**	177647	181677	175072	173804	176032	175041
6×6 (2)	184690	**182331**	200069	200990	184142	**182331**	**182331**	**182331**
6×6 (3)	179777	**174519**	175620	186481	175500	175564	176994	176148
8×8 (1)	330096	323459	316328	375103	316373	313336	**312395**	312558
8×8 (2)	328050	307352	301833	351505	299616	**292667**	294391	297560
8×8 (3)	296379	279625	271637	407457	270830	265853	**265792**	268366
10×10 (1)	426770	420215	404447	514504	402376	**390489**	391025	396456
10×10 (2)	386613	386031	371091	468118	369979	**362574**	364354	366568
10×10 (3)	403731	397800	382180	514514	383321	**378033**	378926	381725
# BRF	3	6	3	3	3	7	6	4

Table 3
insp-QCGA-*inv* vs. PBIL: best, worst, mean, STD, # hits, execution times and statistical tests.

		Network/Algorithm	Best	Worst	Mean	STD	# Hits	Time	H_0	PH
4×4 cells	**1**	***insp*-QCGA-*inv***	98535	98535	**98535.000**	0.000	**30**	290.792	+	**30.5000**
		PBIL	98535	98535	**98535.000**	0.000	**30**	**257.115**		**30.5000**
	2	***insp*-QCGA-*inv***	97156	97156	**97156.000**	0.000	**30**	**285.485**	+	**30.5000**
		PBIL	97156	97156	**97156.000**	0.000	**30**	295.299		**30.5000**
	3	***insp*-QCGA-*inv***	95038	95038	**95038.000**	0.000	**30**	**280.810**	+	**30.5000**
		PBIL	95038	95038	**95038.000**	0.000	**30**	316.151		**30.5000**
6×6 cells	**1**	***insp*-QCGA-*inv***	173701	181802	177182.933	2190.344	1	**1045.954**	–	40.1333
		PBIL	173701	180460	**174799.600**	1607.789	**16**	1174.841		**20.8667**
	2	***insp*-QCGA-*inv***	182331	187916	184069.733	1392.541	5	**1055.617**	–	42.0833
		PBIL	182331	189277	**182562.533**	1268.160	**29**	1149.702		**18.9167**
	3	***insp*-QCGA-*inv***	174519	178379	176924.567	1057.265	1	**1048.666**	–	36.1000
		PBIL	174519	190973	**176505.000**	3150.971	**13**	1132.483		**24.9000**
8×8 cells	**1**	***insp*-QCGA-*inv***	323459	331233	327702.133	2109.068	0	2957.687	–	45.4000
		PBIL	309293	324726	**314227.233**	2784.616	0	**2940.308**		**15.6000**
	2	***insp*-QCGA-*inv***	307352	321290	315170.533	3073.046	0	2944.338	–	45.5000
		PBIL	287149	301897	**294481.900**	5168.672	**7**	**2898.605**		**15.5000**
	3	***insp*-QCGA-*inv***	279625	292276	286772.033	2824.291	0	2931.560	–	45.3667
		PBIL	264204	282511	**268723.900**	4602.308	**1**	**2840.802**		**15.6333**
10×10 cells	**1**	***insp*-QCGA-*inv***	420215	435080	429226.767	4055.418	0	6762.834	–	45.5000
		PBIL	390916	414618	**400901.867**	6329.485	0	**5682.949**		**15.5000**
	2	***insp*-QCGA-*inv***	386031	397478	391784.367	2382.372	0	6716.772	–	45.4667
		PBIL	362858	387087	**370709.133**	6536.299	0	**5515.569**		**15.5333**
	3	***insp*-QCGA-*inv***	397800	405670	403243.533	1878.429	0	6569.924	–	45.5000
		PBIL	375133	395715	**384160.033**	4311.814	0	**5587.166**		**15.5000**

a metric during this comparison. Table 2 presents results of the comparison between the *insp*-QCGA-* approaches and the previously-cited algorithms, where the best results are highlighted in bold and the number of Best Result Found (BRF) is reported. The latter represents how many times each solver could attain the best result obtained in the literature.

As it can be seen, the *insp*-QCGA-*inv* could "*outperform*"/"*be as good as*" the *insp*-QCGA-*orig* and all state-of-the-art algorithms in six instances (networks 1–3 of sizes 4×4 and 6×6 cells). For the remaining instances, no algorithm could outperform the *insp*-QCGA-*inv* in all networks. Indeed, for each network, a different solver outperforms the other algorithms, whereas the *insp*-QCGA-*inv* is always displaying very close results to the best algorithm. Also, it is worth stating that the rest of the study was conducted on the *insp*-QCGA-*inv* since it is the best quantum-inspired algorithm devised in this work.

As indicated in Section 3, the *insp*-QCGA-*inv* and *insp*-QCGA-*orig* differs on the type of rotation gate they use. So, any difference in their performances can hypothetically be brought to a difference in the way their respective rotation gates influence the search process. The results in Table 2 show that the fitness value reached by the *insp*-QCGA-*inv* and *insp*-QCGA-*orig* is similar when tackling networks of size 4×4 cells. Nonetheless, the difference in their fitness values is of the order of 10^3 when addressing networks of size 6×6 cells, and 10^4 when solving instances of 8×8 cells. For the largest networks of size 10×10 cells, the fitness difference is of a matter of $\leqslant 10^3$. This points that the scalability advantage of *insp*-QCGA-*inv* over *insp*-QCGA-*orig* is not exponentially increasing according to the network's size. Also, it indicates that in the case of the MMP, the shortcoming of the original rotation gate R_{insp} (see Eq. (18)) used in the *insp*-QCGA-*orig* might be caused by the problem's fitness landscape type (e.g. multimodality, funnels, etc. [28]) and not only its size. Few can be said about which landscape feature(s) is(are) responsible for this because, as far as the authors' knowledge, no work studied the MMP's landscape yet. When looking back at Fig. 8(b), it can be also thought that the advantage of the *inverted* rotation gate used in *insp*-QCGA-*inv* (see Eq. (19)) stands in its tendency of promoting solutions containing more 1s than 0s throughout the iterations, which suited the MMP instances of size $\geqslant$ 36 cells.

Based on the results in Table 2, the PBIL is the only algorithm capable of outperforming the *insp*-QCGA-*inv*. So, for further understanding of the efficacy of both algorithms, the PBIL and the *insp*-QCGA-*inv* are run using the same settings introduced in Section 4.2. Also, a wide set of information is recorded to assess both solvers on the basis of multiple metrics such as the `Best`, `Worst`, `Mean`, `STD`, `# Hits`, `Time`, H_0 and `PH`. All of these results can be seen in Table 3.

Taking into consideration the metric `Best` in Table 3, it can be seen that *insp*-QCGA-*inv* and PBIL attain equal results in 6 out of 12 instances (networks 1–3 of sizes 4×4 and 6×6 cells). When analysing the results according to metric `Mean`, both algorithms achieve similar results in 3 out of 12 instances (networks 1–3 of size 4×4 cells), while for the remaining instances, the PBIL outperforms the *insp*-QCGA-*inv*. Now, when taking into account the metric `# Hits`, both algorithms attain the same results when tackling 7 out of 12 instances (network 1 of size 8×8 cells and networks 1–3 of sizes 4×4 and 10×10 cells). However, the PBIL outperformed the *insp*-QCGA-*inv* when addressing the remaining instances. The results of statistical tests within the column `PH` confirm those of the classical `Mean` that point out the dissimilarities and similarities between the performances of both the PBIL and *insp*-QCGA-*inv*. Another interesting fact that could be noticed is that the main difference between the achievements of *insp*-QCGA-*inv* and PBIL is their scalability. In fact, it can be seen from Table 3 that for the smallest size of networks (4×4 cells), both PBIL and *insp*-QCGA-*inv* display similar efficiency. Then, as the size of the networks increases (6×6 cells and so on), the difference between the efficiency of the two solvers starts to increase as well. The fitness values' difference when tackling networks of 6×6 cells is of the order of 10^3, then it increases to 10^4 when solving instances of 8×8 cells. The greatness of the difference remains 10^4 when addressing networks of 10×10 cells. This fact supports a part of the previously-made hypothesis stating that, in some cases, the hardness of the MMP comes from the problem fitness landscape type and not only its size.

Regarding the `STD` metric, it can be observed that for networks 1–3 of size 4×4 cells, both the PBIL and *insp*-QCGA-*inv* have equal values which reflect that they are having similar stability of search process. Moving to instances 1 and 2 of size 6×6 cells, the PBIL's `STD` value is smaller than the one of *insp*-QCGA-*inv*, which indicates that the PBIL's search dynamics is more stable. In the remaining 7 networks (network 3 of size 6×6 cells, networks 1–3 of sizes 8×8 and 10×10 cells), the opposite scenario is shown, which points that for networks of larger sizes, the *insp*-QCGA-*inv* has a search process that is more stable than the one of the PBIL. It is to be mentioned also that the *insp*-QCGA-*inv* has an `STD` value that is not correlated to the size of the network being tackled, which indicates that its search process stability is more influenced by the MMP landscape features rather than its size only. It can be also deduced that the MMP's features are affecting the *insp*-QCGA-*inv*'s efficiency and not its search stability. Again, no specific feature(s) can be pointed since, to the best of the authors' knowledge, no work has already researched the MMP's fitness landscape.

With regard to the metric `Time` of Table 3, in 5 out of 12 instances (networks 2 and 3 of size 4×4 cells and networks 1–3 of size 6×6 cells), the *insp*-QCGA-*inv* has a quicker execution than the PBIL. Nonetheless, the PBIL has a faster execution in the 7 remaining instances. This being said, it must be remembered again that the goal here is to offer a study so that advances can be done in the quantum domain, and not to propose the best algorithm that can be built in a quantum computer. The aim is to understand and extract key findings needed for devising better future quantum algorithms, and this by comparing solvers in the three quantum realms (inspired, simulated and real).

Since the PBIL and the *insp*-QCGA-*inv* are two different algorithms, many other (but not firm) hypotheses can be stated about the factors that produce the difference between their results (e.g. the population structure, type of mapping, search operators, etc.). It is worth stating that investigating such hypotheses go beyond the scope of this work. This being said, for further understanding of the behaviour and efficiency of the *insp*-QCGA-*inv*, a thorough parameters' sensitivity analysis and discussion were conducted and are made available in (https://github.com/Zakaria-Dahi/QCGA.git).

4.3.2. Results using a quantum computer simulator

In this section, a second group of experiments is performed in order to answer the first part of the first research question (see `Q1.I`, Section 1) and this by investigating if quantum-inspiration produces results and effects that are enough close to those of well-founded quantum simulators. This is done using the IBMQ simulator, where the experiments presented in this section are those of the best MMP solver found in Section 4.3.1 which is the PBIL and the best *insp*-QCGA-* algorithm found so far, which is the *insp*-QCGA-*inv* and the implementation of the *insp*-QCGA-*inv* using the IBMQ quantum simulator (*sim*-QCGA) (see Fig. 12(a)).

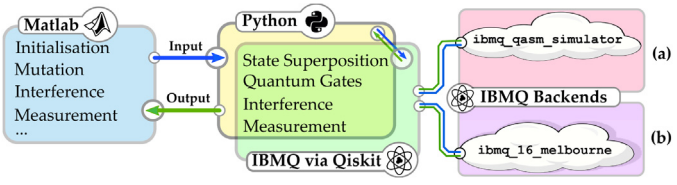

Fig. 12. Execution workflow: (a) *sim*-QCGA and (b) *real*-QCGA.

Considering the limitations of the IBMQ simulator which allows only a maximum of 32 qubits, the experiments will be performed only using networks 1–3 of size 4×4 cells that require 16 qubits instead of the remaining instances (6×6, 8×8 and 10×10 cells) that require $\geqslant$ 36 qubits. Table 4 presents the results of those experiments, while Figs. 13–15 display the fitness evolution of the PBIL, *insp*-QCGA-*inv* and the *sim*-QCGA during one execution. It is worth stating that the results used to create those figures have been chosen on purpose to display the behaviour of the three algorithms when reaching similar results. In addition, considering the assumptions made in Section 4.2 on the non-normality of the results' samples and also regarding the type of the dependent variables being measured, the Kruskal–Wallis test is the one used in these experiments.

As it can be seen from the results in Table 4, the PBIL, *insp*-QCGA-*inv* and the *sim*-QCGA are displaying similar efficiency on the basis of the metrics `Best`, `Worst`, `Mean`, `STD`, `# Hits` or statistically speaking based on the metrics H_0 and `PH`. Also, it can be observed that the *sim*-QCGA is the one taking more time to accomplish one execution. This can be explained by the fact that the procedural execution of the experiments when using the IBMQ quantum simulator involves communications between Matlab and Python that induce substantial processing (see Fig. 12(a)). When looking at Figs. 13–15, a clear difference can be noted in the behaviours of the three algorithms. Taking into consideration the *sim*-QCGA and the *insp*-QCGA-*inv* since they are the main focus of this section, it can be observed that, even if they are using the same working mechanism (see Section 3), they display different

Table 4
Results of the comparison of *sim*-QCGA vs. *insp*-QCGA-*inv* vs. PBIL.

Network	Algorithm	Best	Worst	Mean	STD	# Hits	Time	H_0	PH
1	***sim*-QCGA**	98535	98535	**98535**	0	**30**	14130.909	+	45.50
	insp*-QCGA-*inv	98535	98535	**98535**	0	**30**	290.792	+	45.50
	PBIL	98535	98535	**98535**	0	**30**	**257.115**	+	45.50
2	***sim*-QCGA**	97156	97156	**97156**	0	**30**	14123.787	+	45.50
	insp*-QCGA-*inv	97156	97156	**97156**	0	**30**	**285.485**	+	45.50
	PBIL	97156	97156	**97156**	0	**30**	295.299	+	45.50
3	***sim*-QCGA**	95038	95038	**95038**	0	**30**	14102.020	+	45.50
	insp*-QCGA-*inv	95038	95038	**95038**	0	**30**	**280.810**	+	45.50
	PBIL	95038	95038	**95038**	0	**30**	316.151	+	45.50

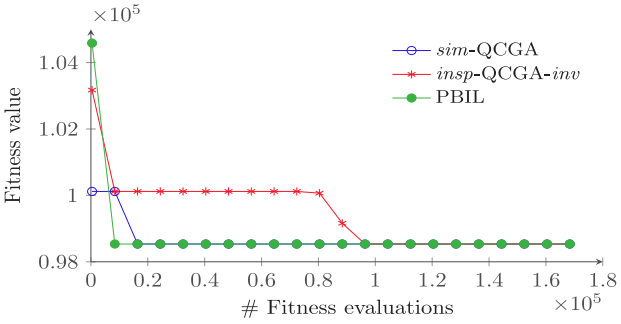

Fig. 13. Network 1 of size 4 × 4 cells.

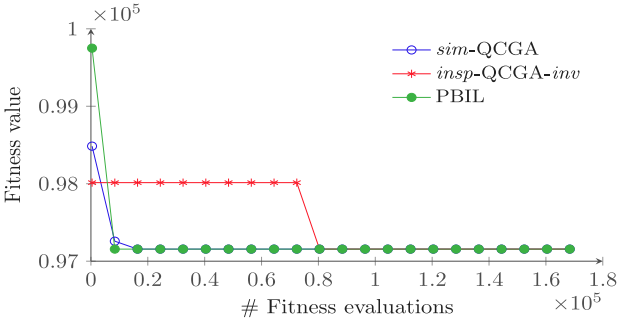

Fig. 14. Network 2 of size 4 × 4 cells.

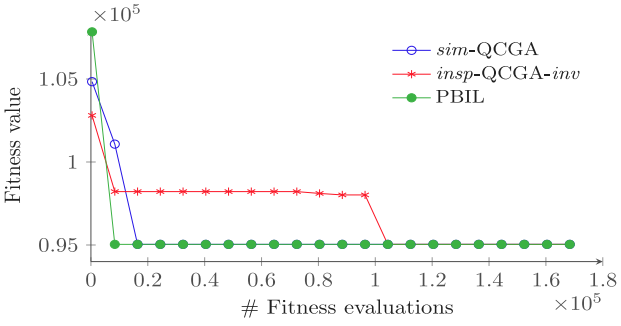

Fig. 15. Network 3 of size 4 × 4 cells.

behaviours, where the *sim*-QCGA has a quicker convergence than the *insp*-QCGA-*inv*. The former tends to converge approximately between $(0.8 \cdot 10^5)$ and $(1.01 \cdot 10^5)$ fitness evaluations in the three networks.

Generally, one of the factors that could influence the convergence rate of a population-based metaheuristic is the composition of the population achieved throughout iterations via the solver's search process. The latter is ruled by the search operators it is composed of. Considering these facts and looking back at both *sim*-QCGA and *insp*-QCGA-*inv* workflows (see Algorithms 2 and 3) and settings (see Section 4.2), it can be seen that both algorithms have the same search processes and tuning. The difference is that the *insp*-QCGA-*inv* performs the initialisation, interference and measurement without a quantum simulator, while the *sim*-QCGA uses one to do it. So, the difference between their convergences could hypothetically be brought to the fact that performing these phases with or without simulator influences differently the algorithms' search dynamics, the population composition throughout the iterations and ultimately the algorithms' convergence rates. Technically speaking, the difference between inspired and simulated quantum operators can be supported by the discussion made in Section 2.2.2. Taking the *insp*-QCGA-*inv*, the parameters' setting has a major impact also on the convergence/efficiency (see Appendix at: https://github.com/Zakaria-Dahi/QCGA.git).

Considering the results in Table 4 and Figs. 13–15, it can be stated that the limitation of the maximum number of qubits allowed by IBMQ quantum simulator allows only tackling small or average-sized problems, but this does not prevent from making some conclusions about the effect of quantum phenomena on the algorithms' behaviours. Thus, as an answer to the first part of the first research question on whether quantum inspiration can replace quantum simulation, it can be said that in the case study presented in this work, the quantum inspiration does not reproduce the same effects of quantum simulators and therefore, they cannot replace them. Indeed, it turns out that the *sim*-QCGA executed on a quantum simulator converges faster than the quantum-inspired solver *insp*-QCGA-*inv*.

4.3.3. Results using a real quantum machine

This third set of experiments aims to answer the second part of the first research question (see `Q1.II`, Section 1) and this by investigating if quantum inspiration produces the same results and effects that real quantum machines achieve. This is done by performing experiments on the proposed *real*-QCGA. The latter is obtained by implementing the *insp-QCGA-inv* on a real quantum machine. As explained in Section 3.7.1, the experiments are done on the largest quantum machine made freely-available by IBM. It is a 15-qubits machine called `ibmq_16_melbourne`. Fig. 12(b) illustrates the sequence of procedural calls to execute the experiments on this IBMQ quantum machine.

It is important to bear in mind also that even if `ibmq_16_melbourne` quantum machine is freely-available, it is subject to several limitations such as `(I)` the number of qubits accessible which is limited only to 15 qubits, `(II)` 8192 of allowed shots and 75 of allowed circuits per job, and `(III)` the time of execution of quantum circuits that is subject to the IBMQ fair-share queuing algorithm. Considering the latter and the number of quantum circuits queued for execution, retrieving results of the execution might take an unpredictable time that varies from seconds to days.

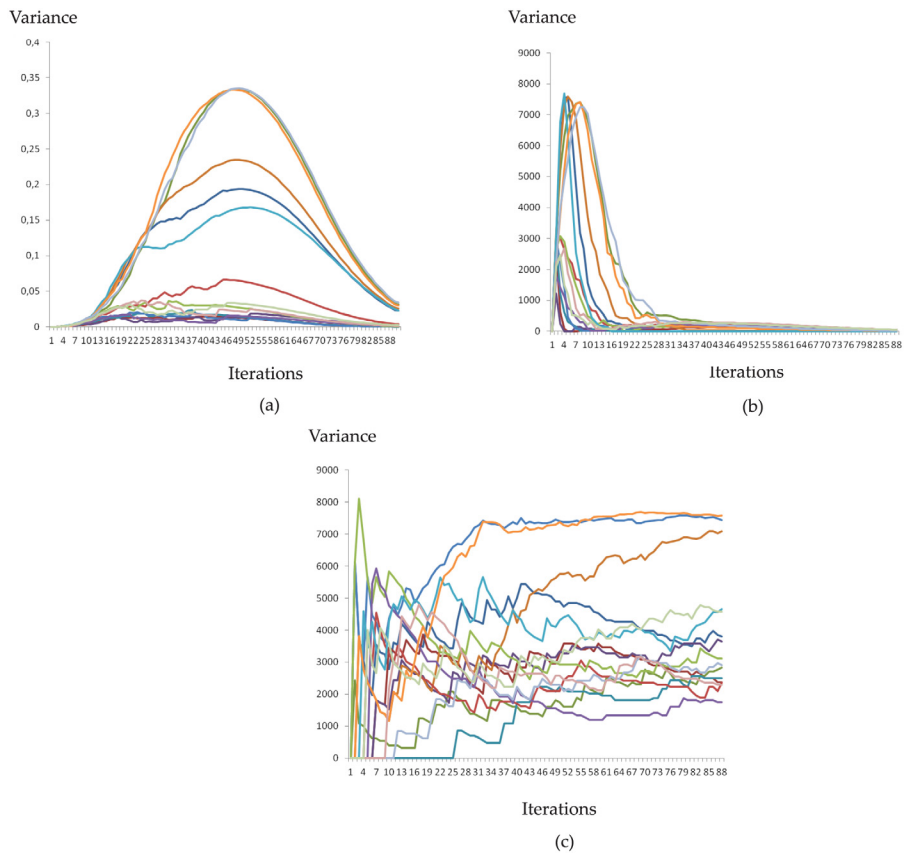

Fig. 16. The variance value evolution: (a) *insp*-QCGA-*inv*, (b) *sim*-QCGA and (c) *real*-QCGA.

Considering the above-indicated constraints, in this work (I) a new instance of 15 cells for the MMP has been created. This new instance is inspired from 6-months realistic communication records accessible in [29]. The data and also the neighbourhood of the newly-created network are made available at [1]. (II) Also, regarding the computational limitations of the `ibmq_16_melbourne` quantum machine, it has been decided to perform experiments until reaching a stop criterion of 70000 fitness evaluations ($88 \leqslant$ iterations $\leqslant 89$) during only *one* execution. The aim here is to study the influence of the use of real quantum machines with real quantum features on the behaviour (e.g. convergence rate, population diversity, etc.) of the *real*-QCGA compared to its simulated and inspired versions. Also, in all the three studied algorithms, the angle θ of the rotation gate has been set to $\frac{\pi}{89}$. This has been done to give a chance to each qubit to reach a probability of 1 to be either in the state $|1>$ or $|0>$ (see Fig. 6(b)).

In these experiments, a comparison will be made between the *real*-QCGA, the *sim*-QCGA and the *insp*-QCGA-*inv*. Fig. 17 and Fig. 16 display the fitness value and the population's variance evolution of all the studied variants, where each variable is represented by a unique colour. The variance has been chosen on purpose since it is a good indicator of how spread the individuals are. Technically speaking, it will indicate how much the variable is far from the mean value of that same variable in all the individuals. This can reflect the behaviour of the population within the *insp*-QCGA-*inv*, *sim*-QCGA and *real*-QCGA. It is also worth stating that the IBMQ quantum machine used to run the *real*-QCGA is accessible worldwide and experiments are executed according to a queue following a priority that depends on the user launching the task (e.g. IBMQ network). Considering these facts, no execution time comparison has been done in this part of the work since its fairness might be compromised by various factors that go beyond the authors' control (e.g. the task's load and priority).

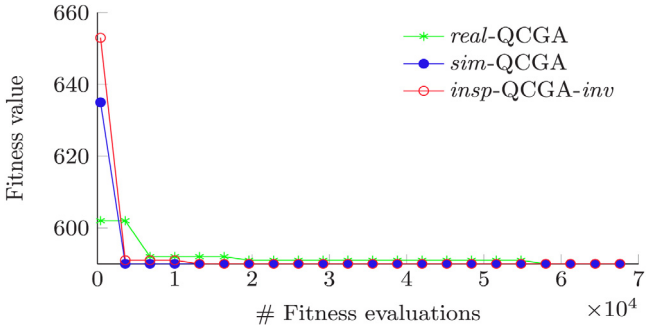

Fig. 17. Network of 15 cells: fitness evolution of *real*-QCGA, *sim*-QCGA and *insp*-QCGA-*inv*.

As it can be seen from Fig. 17, few can be said about the fitness evolution of the three algorithms especially that the *real*-QCGA, *sim*-QCGA and *insp*-QCGA-*inv* reach the same fitness value of 590. On the other hand, according to Fig. 16, a clear difference can be noted in the shape and value of the variance evolution. Indeed, both the *insp*-QCGA-*inv* and *sim*-QCGA have a bell-like variance plot, which shows that all the variables evolve in the same way, but with different amplitudes. On the other hand, the *real*-QCGA has a different variance evolution in terms of shape and value, which indicates that each variable evolves independently of the others. As explained in Section 4.3.2, this might be

[1] The 15-cells network: https://github.com/Zakaria-Dahi/QCGA.git.

caused by being the initialisation, interference and measurement done using a classical machine (without simulator), a quantum simulator or a real quantum machine. In small-sized networks, this does not seem to create a big difference in the efficiency of the three algorithms. However, it might happen that at a large scale, when tackling high-dimensional networks (i.e. problems), such difference in the population convergence might create a difference in the algorithms' efficiency. So, it can be stated that, in this case study, the inspired quantum metaheuristics do not seem to have the same population convergence as real quantum ones. Nonetheless, to confirm and generalise such observation, further experiments need to be performed on larger problems using other comparison criteria (e.g. state vectors). For the time present, this cannot be done until real discrete-gate-based quantum machines with a greater number of qubits are made freely available.

Another observation to be made is that the variance evolution in Figs. 16(a) and (b) supports the statement made in Section 4.3.2 about the causes of the difference between the convergence of *sim*-QCGA and *insp*-QCGA-*inv*. Indeed, the variance evolution of the *sim*-QCGA looks more compact, which indicates that the population converges quickly to a final state. On the other hand, the population's variance in *insp*-QCGA-*inv* is more spread and continue evolving longer, which points that the population takes more time to converge to a final state. The conclusion that can be made in this work is that the type of machine used to execute quantum-related phases has a noticeable effect on the convergence of the algorithm and its population. The new fact to be kept in mind is that performing the initialisation, interference and measurement on quantum simulator speeds up the population convergence, but keeps the same type of evolution (i.e. bell-like) as when using classical machines. Contrastingly, performing the quantum-related phases on a real quantum machine changes both the shape and value of the population convergence.

4.4. The applications and restrictions of the proposed quantum cellular genetic algorithms

The findings and results given in Sections 4.3.1–4.3.3 could have several applications such as (I) understanding the MMP fitness landscape and building efficient and problem-aware optimisers to solve this problem (e.g. grey-box [30]), (II) extracting the strength and weaknesses of *real*-, *sim*- and *insp*-QCGA-* metaheuristics, (III) deciding whether these algorithms are suitable (or not) for a given problem (e.g. via search trajectory analysis [31]), and if needed adapt them to the problem being tackled, (IV) understanding how inspired, simulated and real quantum metaheuristics behave and how quantum operators and hardware influence their search processes, (V) deciding which type of quantum hardware is adequate for a given solver/problem, etc. The approaches proposed in this work including the *insp*-QCGA-*, *sim*-QCGA and *real*-QCGA could be applied to solve numerous complex optimisation problems in various theoretical and practical fields including the quantum computing domain itself. Enumerating all the possible applications would be too substantial, but some examples can be cited such as: artificial intelligence (e.g. machine learning [32]), industry/logistics (e.g. transportation, packaging, etc. [33]), medicine (e.g. resources' allocation [34]), bio-informatics (e.g. deoxyribonucleic acid fragment assembly [35]), ecology/renewable energies (e.g. wind turbine design [36]), smart cities (e.g. electric vehicles' charging [37]), communication networks (e.g. vehicular networks [38]) and quantum computing [39]. For further applications, it can be referred to [40].

This being said, despite having a wide range of applications, some approaches proposed here could have, in a given context, some (I) *applicability*, (II) *execution time* or (III) *scalability* constraints. First, considering the solution representation used in the proposal, all the devised approaches can be applied, without any additional processes, for solving binary problems. They could be also used to solve other types of problems (e.g. continuous, integer, etc.), but further mapping processes would be needed to convert binary solutions into an adequate format. This limitation could be solved in different ways such as using a grey-coded solution representation or mapping functions [10]. Secondly, in the case of the *real*-QCGA, this algorithm is run on a real IBMQ quantum machine. The execution flow of the tasks in that machine is ruled by a queuing protocol, where members of the *IBMQ network* are given top priority for running their programs. So, if the *real*-QCGA is executed by a user out of the IBMQ network, the execution of the devised algorithm could take longer. As a possible solution to this limitation, the *real*-QCGA can be run on other quantum machines (e.g. IonQ) or the IBMQ machines by joining the IBMQ network. Thirdly, the applicability of the *sim*-QCGA and *real*-QCGA for solving large-scale problems depends on the number of qubits available on the simulator/quantum machine being used. As possible solution to this restriction, the *sim*-QCGA and *real*-QCGA can be run by splitting the quantum circuits using the hybrid Schrödinger–Feynman algorithm like it was done in [5] and execute each sub-circuit independently, then merge the sub-circuits back again. Based on the above-stated facts, it can be noted that all the possible limitations that the approaches devised in this work might face are *hardware* ones considering that quantum machines are still a new technology being built and few large-scale and open-access quantum simulators and machines are available nowadays. The above-stated solutions provided to these limitations are at *zero* cost. So, the user will not have to pay any fees to execute the proposed approaches if he/she runs into such limitations. On the other hand, if some fees can be afforded, all the above-mentioned limitations could possibly be solved by just executing the proposed approaches on a commercial quantum computer.

4.5. Guidelines to design future quantum metaheuristics

In this section, both parts of the second research question are answered (see Q2, Section 1) to establish whether (I) considering the substantial speed-up provided by quantum computers, will quantum-inspired metaheuristics, with their present design, still be relevant and (II) if no, how should they be redesigned. Based on the results presented in the previous sections, it can be seen that the quantum-inspired algorithms do not usually reflect the real behaviour of quantum metaheuristics when using real quantum machines. Indeed, considering the way quantum-inspired metaheuristics are designed and how quantum principles are included in them, several design aspects might need to be rethought to make such techniques fully quantum.

First, to be executed on a discrete variable gate-model quantum system, an algorithm has to be written in the form of a quantum circuit (i.e. a set of quantum and classical registers composed of quantum bits and altered by quantum gates). This could help take profit from the computation speed-up induced by quantum machines. Thus, real quantum algorithms are mostly "quantum circuits" and not algorithms in the sense of classical computer science like it is done in quantum-inspired metaheuristics. This can be noted in Grover's search [1] and Shor's factoring algorithms [14]. Taking the latter, each quantum circuit is designed for a specific task/problem. So, the problem-dependence/independence in metaheuristics might be tricky to reproduce on quantum machines.

Secondly, the way quantum principles are emulated into quantum-inspired metaheuristics is often simplistically done in most of the existing literature. Indeed, when taking the superposition of states as an example, it stands in using simple real numbers to represent probabilities to be 0 or 1. When considering the interference, it can be seen as a matrix operation on a flat basis. Concerning these details, an observation-triggering fact is that all of these are already done in some distribution-based metaheuristics. So technically speaking, this makes that some quantum-inspired metaheuristics are a kind of estimation of distribution-based algorithms.

5. Conclusions and future work

Quantum-inspired metaheuristics are optimisation algorithms that integrate some principles inspired from quantum mechanics into classical metaheuristics. Due to the difference between the quantum features being inspired from and the classical non-quantum machines they are run on, several important questions come to light such as (Q1) are quantum principles integrated into quantum-inspired algorithms in such a way that guarantees their functioning on real or simulated quantum computers? (Q2) How quantum metaheuristics should be designed in order to take full advantage of the computational speed-up provided by quantum devices? Answering these questions is critical since it will allow building operational quantum algorithms that take full profit from quantum machines' calculation acceleration.

In this paper, the above-stated questions have been answered by proposing and analysing (I) a quantum-*inspired* and, (II) for the first time in the literature, both a quantum-*simulated* and a *real* quantum cellular genetic algorithm. This has been done using classical machines, a 32-qubits IBMQ quantum simulator and a 15-qubits IBMQ real quantum machine. The proposals' performances (e.g. efficiency, scalability, convergence, etc.) have been thoroughly assessed by solving the NP-hard users' MMP in advanced cellular networks. Furthermore, various instances and comparisons have been made against a wide range of solvers.

The experimentation's outcomes have shown that (I) quantum-inspired metaheuristics, in their current form, do not provide significant efficiency enhancements compared to classical MMP solvers. This can be explained by the fact that their present design and implementation made them not quantum-material algorithms, but more related to classical distribution-based algorithms rather than quantum algorithms. (II) It has been found that inspiration like it is done could not result in the same outcomes as simulation or real quantum machines do. Indeed, the current design of quantum-inspired algorithms makes that the resulting behaviour of the produced techniques is different from the one implemented on quantum simulators or real quantum machines. Thus, the findings that result from the use of quantum-inspired techniques might be difficult to reproduce on simulators or real quantum systems. (III) It is believed that quantum search algorithms should be designed in terms of quantum circuits to benefit from the computational speed-up that quantum machines offer. This encourages investigating new research lines in quantum algorithms and software design.

As a perspective, plans are set to extend the present study using a real quantum computer with a larger number of qubits. Also, it is intended to study the influence of heterogeneous quantum platforms/machines (e.g. Google playground, Amazon Braket, D-Wave, etc.) on the behaviour of quantum solvers. Finally, efforts will also be made to study other metaheuristics in the quantum realm.

CRediT authorship contribution statement

Zakaria Abdelmoiz Dahi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Enrique Alba:** Validation, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: The authors acknowledge that this research is partially funded by the Universidad de Málaga, Consejería de Economía y Conocimiento de la Junta de Andalucía and FEDER under grant number UMA18-FEDERJA-003 (PRECOG); under grant PID 2020-116727RB-I00 (HUmove) funded by MCIN/AEI/10.13039/501100011033; and TAILOR ICT-48 Network (No 952215) funded by EU Horizon 2020 research and innovation programme. Funding for open access charge is supported by the Universidad de Málaga/CBUA. The authors also acknowledge the use of the IBMQ for this work. The views expressed are purely those of the authors and do not reflect the official policy or position of IBM, the IBMQ team or the European Commission. Finally, the authors also acknowledge that an instance studied in this work was inspired from the CRAWDAD dataset spitz/cellular.

Acknowledgements

Authors acknowledge that this research is partially funded by the Universidad de Málaga, Consejería de Economía *y* Conocimiento de la Junta de Andalucía and FEDER under grant number UMA18-FEDERJA-003 (PRECOG); under grant PID 2020-116727RB-I00 (HUmove) funded by MCIN/AEI/10.13039/501100011033; and TAILOR ICT-48 Network (No 952215) funded by EU Horizon 2020 research and innovation programme. Funding for open access charge is supported by the Universidad de Málaga/CBUA. The authors also acknowledge that an instance studied in this work was inspired from the CRAWDAD dataset spitz/cellular. The authors acknowledge also the use of the IBMQ for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBMQ team. Finally, the authors would like to state that the views expressed are purely those of the writer and may not in any circumstances be regarded as stating an official position of the European Commission.

References

[1] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, in: STOC, vol. 96, Association for Computing Machinery, New York, NY, USA, 1996, pp. 212–219, http://dx.doi.org/10.1145/237814.237866.

[2] M.W. Przewozniczek, M.M. Komarnicki, Empirical linkage learning, IEEE Trans. Evol. Comput. 24 (6) (2020) 1097–1111, http://dx.doi.org/10.1109/TEVC.2020.2985497.

[3] L. Mu, P. Wang, G. Xin, Quantum-inspired algorithm with fitness landscape approximation in reduced dimensional spaces for numerical function optimization, Inform. Sci. 527 (2020) 253–278, http://dx.doi.org/10.1016/j.ins.2020.03.035, URL http://www.sciencedirect.com/science/article/pii/S0020025520302152.

[4] M. Ross, O. H., A review of quantum-inspired metaheuristics: Going from classical computers to real quantum computers, IEEE Access 8 (2020) 814–838, http://dx.doi.org/10.1109/ACCESS.2019.2962155.

[5] F. Arute, K. Arya, R. Babbush, D. Bacon, J. Bardin, R. Barends, R. Biswas, S. Boixo, F. Brandao, D. Buell, B. Burkett, Y. Chen, J. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C.M. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. Harrigan, M. Hartmann, A. Ho, M.R. Hoffmann, T. Huang, T. Humble, S. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. Klimov, S. Knysh, A. Korotkov,

F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J.R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M.Y. Niu, E. Ostby, A. Petukhov, J. Platt, C. Quintana, E.G. Rieffel, P. Roushan, N. Rubin, D. Sank, K.J. Satzinger, V. Smelyanskiy, K.J. Sung, M. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z.J. Yao, P. Yeh, A. Zalcman, H. Neven, J. Martinis, Quantum supremacy using a programmable superconducting processor, Nature 574 (2019) 505–510, URL https://www.nature.com/articles/s41586-019-1666-5.

[6] E. Pednault, J.A. Gunnels, G. Nannicini, L. Horesh, R. Wisnieff, Leveraging secondary storage to simulate deep 54-qubit sycamore circuits, 2019, http://arxiv.org/abs/1910.09534 [arXiv:1910.09534].

[7] Y.A. Liu, X.L. Liu, F.N. Li, H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen, C. Guo, H. Huang, W. Wu, D. Chen, Closing the "quantum supremacy" gap: achieving real-time simulation of a random quantum circuit using a new sunway supercomputer, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, in: SC '21, Association for Computing Machinery, New York, NY, USA, 2021, http://dx.doi.org/10.1145/3458817.3487399, https://doi.org/10.1145/3458817.3487399.

[8] E.-G. Talbi, Metaheuristics: From Design To Implementation, Wiley Publishing, 2009.

[9] D. Whitley, Next generation genetic algorithms: a user's guide and tutorial, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Springer International Publishing, Cham, 2019, pp. 245–274, http://dx.doi.org/10.1007/978-3-319-91086-4_8.

[10] Z.A. Dahi, C. Mezioud, A. Draa, A 0-1 bat algorithm for cellular network optimisation: a systematic study on mapping techniques, Int. J. Reason.-Based Intell. Syst. 9 (2017) 22–42, http://dx.doi.org/10.1504/IJRIS.2017.10007147, URL https://doi.org/10.1504/IJRIS.2017.10007147.

[11] T. Harada, E. Alba, Parallel genetic algorithms: A useful survey, ACM Comput. Surv. 53 (4) (2020) http://dx.doi.org/10.1145/3400031.

[12] C. Salto, E. Alba, Cellular genetic algorithms: Understanding the behavior of using neighborhoods, Appl. Artif. Intell. 33 (10) (2019) 863–880, http://dx.doi.org/10.1080/08839514.2019.1646005.

[13] Z.A. Dahi, E. Alba, The grid-to-neighbourhood relationship in cellular GAs: from design to solving complex problems, Soft Comput. J. 24 (5) (2019) 3569–3589.

[14] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. 26 (5) (1997) 1484–1509, http://dx.doi.org/10.1137/s0097539795293172, URL http://dx.doi.org/10.1137/S0097539795293172.

[15] A. Narayanan, M. Moore, Quantum-inspired genetic algorithms, in: Proceedings of the International Conference on Evolutionary Computation, in: CEC, vol. 96, IEEE, 1995, pp. 61–66.

[16] P. Sadowski, E. Alba, Design and implementation of genetic algorithms in actual quantum devices: The annealing paradigm on DWAVE, 2021, http://dx.doi.org/10.5281/zenodo.4737389, Zenodo.

[17] G. de Andoin Mikel, E. Javier, Implementable hybrid quantum ant colony optimization algorithm, 2021, http://arxiv.org/abs/2107.03845 [arXiv:2107.03845].

[18] M. Ghosh, N. Dey, D. Mitra, A. Chakrabarti, A novel quantum algorithm for ant colony optimization, 2021, http://arxiv.org/abs/2010.07413 [arXiv:2010.07413].

[19] Z.A. Dahi, E. Alba, A. Draa, A stop-and-start adaptive cellular genetic algorithm for mobility management of GSM-LTE cellular network users, Expert Syst. Appl. 106 (2018) 290–304.

[20] S. Katoch, S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, Multimed Tools Appl. 80 (2021) 8091–8126.

[21] D. Konar, S. Bhattacharyya, T.K. Gandhi, B.K. Panigrahi, A quantum-inspired self-supervised network model for automatic segmentation of brain MR images, Appl. Soft Comput. 93 (2020) 106348, http://dx.doi.org/10.1016/j.asoc.2020.106348, URL http://www.sciencedirect.com/science/article/pii/S156849462030288X.

[22] H. Xiong, Z. Wu, H. Fan, G. Li, G. Jiang, Quantum rotation gate in quantum-inspired evolutionary algorithm: A review, analysis and comparison study, Swarm Evol. Comput. 42 (2018) 43–57, http://dx.doi.org/10.1016/j.swevo.2018.02.020, URL http://www.sciencedirect.com/science/article/pii/S2210650216303807.

[23] Z.A. Dahi, Optimisation Problem Solving In the Field of Cellular Networks (Ph.D. thesis), Constantine 2 University, 2017.

[24] Z.A. Dahi, E. Alba, G. Luque, A takeover time-driven adaptive evolutionary algorithm for mobile user tracking in pre-5g cellular networks, Applied Soft Computing (2021) 107992, http://dx.doi.org/10.1016/j.asoc.2021.107992, https://www.sciencedirect.com/science/article/pii/S1568494621009145.

[25] D.L. González-Álvarez, A. Rubio-Largo, M.A. Vega-Rodrí guez, S.M. Almeida-Luz, J.A. Gómez-Pulido, J.M. Sánchez-Pérez, Solving the reporting cells problem by using a parallel team of evolutionary algorithms, Logic J. IGPL 20 (4) (2011) 722–731, http://dx.doi.org/10.1093/jigpal/jzr016.

[26] V. Berrocal-Plaza, M.A. Vega-Rodríguez, J.M. Sánchez-Pérez, A strength Pareto approach to solve the reporting cells planning problem, in: B. Murgante, S. Misra, A.M.A.C. Rocha, C. Torre, J.G. Rocha, M.I. Falcão, D. Taniar, B.O. Apduhan, O. Gervasi (Eds.), Computational Science and Its Applications, ICCSA 2014, Springer International Publishing, Cham, 2014, pp. 212–223.

[27] MD SAJID ANIS and Héctor Abraham and AduOffei and Rochisha Agarwal and Gabriele Agliardi and Merav Aharoni and Ismail Yunus Akhalwaya and Gadi Aleksandrowicz and Thomas Alexander and Matthew Amy and Sashwat Anagolum and Eli Arbel and Abraham Asfaw and Anish Athalye and Artur Avkhadiev and Carlos Azaustre and PRATHAMESH BHOLE and Abhik Banerjee and Santanu Banerjee and Will Bang and Aman Bansal and Panagiotis Barkoutsos and Ashish Barnawal and George Barron and George S. Barron and Luciano Bello and Yael Ben-Haim and M. Chandler Bennett and Daniel Bevenius and Dhruv Bhatnagar and Arjun Bhobe and Paolo Bianchini and Lev S. Bishop and Carsten Blank and Sorin Bolos and Soham Bopardikar and Samuel Bosch and Sebastian Brandhofer and Brandon and Sergey Bravyi and Nick Bronn and Bryce-Fuller and David Bucher and Artemiy Burov and Fran Cabrera and Padraic Calpin and Lauren Capelluto and Jorge Carballo and Ginés Carrascal and Adam Carriker and Ivan Carvalho and Adrian Chen and Chun-Fu Chen and Edward Chen and Jielun (Chris) Chen and Richard Chen and Franck Chevallier and Kartik Chinda and Rathish Cholarajan and Jerry M. Chow and Spencer Churchill and CisterMoke and Christian Claus and Christian Clauss and Caleb Clothier and Romilly Cocking and Ryan Cocuzzo and Jordan Connor and Filipe Correa and Abigail J. Cross and Andrew W. Cross and Simon Cross and Juan Cruz-Benito and Chris Culver and Antonio D. Córcoles-Gonzales and Navaneeth D and Sean Dague and Tareq El Dandachi and Animesh N Dangwal and Jonathan Daniel and Marcus Daniels and Matthieu Dartiailh and Abdón Rodríguez Davila and Faisal Debouni and Anton Dekusar and Amol Deshmukh and Mohit Deshpande and Delton Ding and Jun Doi and Eli M. Dow and Eric Drechsler and Eugene Dumitrescu and Karel Dumon and Ivan Duran and Kareem EL-Safty and Eric Eastman and Grant Eberle and Amir Ebrahimi and Pieter Eendebak and Daniel Egger and ElePT and Emilio and Alberto Espiricueta and Mark Everitt and Davide Facoetti and Farida and Paco Martín Fernández and Samuele Ferracin and Davide Ferrari and Axel Hernández Ferrera and Romain Fouilland and Albert Frisch and Andreas Fuhrer and Bryce Fuller and MELVIN GEORGE and Julien Gacon and Borja Godoy Gago and Claudio Gambella and Jay M. Gambetta and Adhisha Gammanpila and Luis Garcia and Tanya Garg and Shelly Garion and James R. Garrison and Jim Garrison and Tim Gates and Leron Gil and Austin Gilliam and Aditya Giridharan and Juan Gomez-Mosquera and Gonzalo and Salvador de la Puente González and Jesse Gorzinski and Ian Gould and Donny Greenberg and Dmitry Grinko and Wen Guan and Dani Guijo and John A. Gunnels and Harshit Gupta and Naman Gupta and Jakob M. Günther and Mikael Haglund and Isabel Haide and Ikko Hamamura and Omar Costa Hamido and Frank Harkins and Kevin Hartman and Areeq Hasan and Vojtech Havlicek and Joe Hellmers and Łukasz Herok and Stefan Hillmich and Hiroshi Horii and Connor Howington and Shaohan Hu and Wei Hu and Junye Huang and Rolf Huisman and Haruki Imai and Takashi Imamichi and Kazuaki Ishizaki and Ishwor and Raban Iten and Toshinari Itoko and Alexander Ivrii and Ali Javadi and Ali Javadi-Abhari and Wahaj Javed and Qian Jianhua and Madhav Jivrajani and Kiran Johns and Scott Johnstun and Jonathan-Shoemaker and JosDenmark and JoshDumo and John Judge and Tal Kachmann and Akshay Kale and Naoki Kanazawa and Jessica Kane and Kang-Bae and Annanay Kapila and Anton Karazeev and Paul Kassebaum and Josh Kelso and Scott Kelso and Vismai Khanderao and Spencer King and Yuri Kobayashi and Kovi11Day and Arseny Kovyrshin and Rajiv Krishnakumar and Vivek Krishnan and Kevin Krsulich and Prasad Kumkar and Gawel Kus and Ryan LaRose and Enrique Lacal and Raphaël Lambert and Haggai Landa and John Lapeyre and Joe Latone and Scott Lawrence and Christina Lee and Gushu Li and Jake Lishman and Dennis Liu and Peng Liu and Liam Madden and Yunho Maeng and Saurav Maheshkar and Kahan Majmudar and Aleksei Malyshev and Mohamed El Mandouh and Joshua Manela and Manjula and Jakub Marecek and Manoel Marques and Kunal Marwaha and Dmitri Maslov and Paweł Maszota and Dolph Mathews and Atsushi Matsuo and Farai Mazhandu and Doug McClure and Maureen McElaney and Cameron McGarry and David McKay and Dan McPherson and Srujan Meesala and Dekel Meirom and Corey Mendell and Thomas Metcalfe and Martin Mevissen and Andrew Meyer and Antonio Mezzacapo and Rohit Midha and Daniel Miller and Zlatko Minev and Abby Mitchell and Nikolaj Moll and Alejandro Montanez and Gabriel Monteiro and Michael Duane Mooring and Renier Morales and Niall Moran and David Morcuende and Seif Mostafa and Mario Motta and Romain Moyard and Prakash Murali and Jan Müggenburg and Tristan NEMOZ and David Nadlinger and Ken Nakanishi and Giacomo Nannicini and Paul Nation and Edwin Navarro and Yehuda Naveh and Scott Wyman Neagle and Patrick Neuweiler and Aziz Ngoueya and Johan Nicander and Nick-Singstock and Pradeep Niroula and Hassi Norlen and NuoWenLei and Lee James O'Riordan and Oluwatobi Ogunbayo and Pauline Ollitrault

and Tamiya Onodera and Raul Otaolea and Steven Oud and Dan Padilha and Hanhee Paik and Soham Pal and Yuchen Pang and Ashish Panigrahi and Vincent R. Pascuzzi and Simone Perriello and Eric Peterson and Anna Phan and Francesco Piro and Marco Pistoia and Christophe Piveteau and Julia Plewa and Pierre Pocreau and Alejandro Pozas-Kerstjens and Rafał Pracht and Milos Prokop and Viktor Prutyanov and Sumit Puri and Daniel Puzzuoli and Jesús Pérez and Quant02 and Quintiii and Isha R and Rafey Iqbal Rahman and Arun Raja and Roshan Rajeev and Nipun Ramagiri and Anirudh Rao and Rudy Raymond and Oliver Reardon-Smith and Rafael Martín-Cuevas Redondo and Max Reuter and Julia Rice and Matt Riedemann and Rietesh and Drew Risinger and Marcello La Rocca and Diego M. Rodríguez and RohithKarur and Ben Rosand and Max Rossmannek and Mingi Ryu and Tharrmashastha SAPV and Nahum Rosa Cruz Sa and Arijit Saha and Abdullah Ash- Saki and Sankalp Sanand and Martin Sandberg and Hirmay Sandesara and Ritvik Sapra and Hayk Sargsyan and Aniruddha Sarkar and Ninad Sathaye and Bruno Schmitt and Chris Schnabel and Zachary Schoenfeld and Travis L. Scholten and Eddie Schoute and Mark Schulterbrandt and Joachim Schwarm and James Seaward and Sergi and Ismael Faro Sertage and Kanav Setia and Freya Shah and Nathan Shammah and Rohan Sharma and Yunong Shi and Jonathan Shoemaker and Adenilton Silva and Andrea Simonetto and Deeksha Singh and Divyanshu Singh and Parmeet Singh and Phattharaporn Singkanipa and Yukio Siraichi and Siri and Jesús Sistos and Iskandar Sitdikov and Seyon Sivarajah and Magnus Berg Sletfjerding and John A. Smolin and Mathias Soeken and Igor Olegovich Sokolov and Igor Sokolov and Vicente P. Soloviev and SooluThomas and Starfish and Dominik Steenken and Matt Stypulkoski and Adrien Suau and Shaojun Sun and Kevin J. Sung and Makoto Suwama and Oskar Słowik and Hitomi Takahashi and Tanvesh Takawale and Ivano Tavernelli and Charles Taylor and Pete Taylour and Soolu Thomas and Kevin Tian and Mathieu Tillet and Maddy Tod and Miroslav Tomasik and Caroline Tornow and Enrique de la Torre and Juan Luis Sánchez Toural and Kenso Trabing and Matthew Treinish and Dimitar Trenev and TrishaPe and Felix Truger and Georgios Tsilimigkounakis and Davindra Tulsi and Wes Turner and Yotam Vaknin and Carmen Recio Valcarce and Francois Varchon and Adish Vartak and Almudena Carrera Vazquez and Prajjwal Vijaywargiya and Victor Villar and Bhargav Vishnu and Desiree Vogt-Lee and Christophe Vuillot and James Weaver and Johannes Weidenfeller and Rafal Wieczorek and Jonathan A. Wildstrom and Jessica Wilson and Erick Winston and WinterSoldier and Jack J. Woehr and Stefan Woerner and Ryan Woo and Christopher J. Wood and Ryan Wood and Steve Wood and James Wootton and Matt Wright and Lucy Xing and Jintao YU and Bo Yang and Daniyar Yeralin and Ryota Yonekura and David Yonge-Mallo and Ryuhei Yoshida and Richard Young and Jessie Yu and Lebin Yu and Christopher Zachow and Laura Zdanski and Helena Zhang and Christa Zoufal and aeddins-ibm and alexzhang13 and b63 and bartek-bartlomiej and bcamorrison and brandhsn and charmerDark and deeplokhande and dekel.meirom and dime10 and dlasecki and ehchen and fanizzamarco and fs1132429 and gadial and galeinston and georgezhou20 and georgios-ts and gruu and hhorii and hykavitha and itoko and jessica-angel7 and jezerjojo14 and jliu45 and jscott2 and klinvill and krutik2966 and ma5x and michelle4654 and msuwama and ntgiwsvp and ordmoj and sagar pahwa and pritamsinha2304 and ryancocuzzo and saswati-qiskit and septembrr and sethmerkel and shaashwat and sternparky and strickroman and tigerjack and tsura-crisaldo and vadebayo49 and welien and willhbang and wmurphy-collabstar and yang.luh and Mantas Čepulkovskis, Qiskit: An Open-source Framework for Quantum Computing, 2021, http://dx.doi.org/10.5281/zenodo.2573505.

[28] S. Tari, G. Ochoa, Local search pivoting rules and the landscape global structure, in: Proceedings of the Genetic and Evolutionary Computation Conference, in: GECCO, vol. 21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 278–286, http://dx.doi.org/10.1145/3449639.3459295.

[29] M. S., Crawdad dataset spitz/cellular (v. 2011-05-04), Downloaded from https://crawdad.org/spitz/cellular/20110504 (2011) http://dx.doi.org/10.15783/C71P4C.

[30] A. Bouter, S.C. Maree, T. Alderliesten, P.A.N. Bosman, Leveraging conditional linkage models in gray-box optimization with the real-valued gene-pool optimal mixing evolutionary algorithm, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, in: GECCO, vol. 20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 603–611, http://dx.doi.org/10.1145/3377930.3390225, URL https://doi.org/10.1145/3377930.3390225.

[31] G. Ochoa, K.M. Malan, C. Blum, Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics, Appl. Soft Comput. 109 (2021) 107492, http://dx.doi.org/10.1016/j.asoc.2021.107492, URL https://www.sciencedirect.com/science/article/pii/S1568494621004154.

[32] D. Oliva, E.H. Houssein, S. Hinojosa, Metaheuristics in Machine Learning: Theory and Applications, first ed., Springer International Publishing, 2021.

[33] C. Iliopoulou, K. Kepaptsoglou, E. Vlahogianni, Metaheuristics for the transit route network design problem: a review and comparative analysis, Public Transp. 11 (2019) 487–521.

[34] W. Crown, N. Buyukkaramikli, P. Thokala, A. Morton, M.Y. Sir, D.A. Marshall, J. Tosh, W.V. Padula, M.J. Ijzerman, P.K. Wong, K.S. Pasupathy, Constrained optimization methods in health services research—An introduction: Report 1 of the ISPOR optimization methods emerging good practices task force, Value in Health 20 (3) (2017) 310–319, http://dx.doi.org/10.1016/j.jval.2017.01.013, URL https://www.sciencedirect.com/science/article/pii/S1098301517300827.

[35] Uzma, Z. Halim, Optimizing the DNA fragment assembly using metaheuristic-based overlap layout consensus approach, Appl. Soft Comput. 92 (2020) 106256, http://dx.doi.org/10.1016/j.asoc.2020.106256, URL https://www.sciencedirect.com/science/article/pii/S1568494620301964.

[36] J.X. Vianna Neto, E.J. Guerra Junior, S.R. Moreno, H.V. Hultmann Ayala, V.C. Mariani, L. dos Santos Coelho, Wind turbine blade geometry design based on multi-objective optimization using metaheuristics, Energy 162 (2018) 645–658, http://dx.doi.org/10.1016/j.energy.2018.07.186, URL https://www.sciencedirect.com/science/article/pii/S036054421831483X.

[37] J. García-Álvarez, M.A. González, C.R. Vela, Metaheuristics for solving a real-world electric vehicle charging scheduling problem, Appl. Soft Comput. 65 (2018) 292–306, http://dx.doi.org/10.1016/j.asoc.2018.01.010, URL https://www.sciencedirect.com/science/article/pii/S1568494618300164.

[38] A.D. Masegosa, E. Osaba, J.S. Angarita-Zapata, I. Laña, J.D. Ser, Nature-inspired metaheuristics for optimizing information dissemination in vehicular networks, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, in: GECCO, vol. 19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1312–1320, http://dx.doi.org/10.1145/3319619.3326847.

[39] L. Moro, M.G.A. Paris, M. Restelli, E. Prati, Quantum compiling by deep reinforcement learning, Commun. Phys. 4 (2021) 2399–3650, http://dx.doi.org/10.1038/s42005-021-00684-3, URL https://doi.org/10.1038/s42005-021-00684-3.

[40] N. Khanduja, B. Bhushan, Recent advances and application of metaheuristic algorithms: A survey (2014–2020), in: H. Malik, A. Iqbal, P. Joshi, S. Agrawal, F.I. Bakhsh (Eds.), Metaheuristic and Evolutionary Computation: Algorithms and Applications, Springer Singapore, Singapore, 2021, pp. 207–228, http://dx.doi.org/10.1007/978-981-15-7571-6_10.

Zakaria Abdelmoiz Dahi is a researcher at the University of Malaga (Spain) and also an associate professor at the University of Constantine 2 (Algeria). His research interests include artificial intelligence, quantum computing, and their design/application for solving optimisation problems.

Enrique Alba is a Professor of computer science at the University of Malaga, Spain. His current research interests involve the design and application of metaheuristics and bio-inspired systems to real problems in telecommunications, combinatorial optimisation, software engineering, and smart cities. He is the 5th most influential researcher in computer science in Spain (JCR), the 1st of his university in engineering, and a prolific author according to DBLP, with an H-index of 65 and more than 19000 cites to his works.