Expert Systems With Applications 67 (2017) 71–83

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Quantum inspired evolutionary algorithm for ordering problems

Luciano Reis da Silveira, Ricardo Tanscheit, Marley M.B.R. Vellasco*

Department of Electrical Engineering, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rua Marquês de São Vicente, 225, Rio de Janeiro – RJ, Brazil

ARTICLE INFO

Article history:
Received 2 May 2016
Revised 20 July 2016
Accepted 29 August 2016
Available online 15 September 2016

Keywords:
Quantum inspired evolutionary algorithm
Ordering optimization problem
Quantum bit
Vehicle routing problem

ABSTRACT

This paper proposes a new quantum-inspired evolutionary algorithm for solving ordering problems. Quantum-inspired evolutionary algorithms based on binary and real representations have been previously developed to solve combinatorial and numerical optimization problems, providing better results than classical genetic algorithms with less computational effort. However, for ordering problems, order-based genetic algorithms are more suitable than those with binary and real representations. This is because specialized crossover and mutation processes are employed to always generate feasible solutions. Therefore, this work proposes a new quantum-inspired evolutionary algorithm especially devised for ordering problems (QIEA-O). Two versions of the algorithm have been proposed. The so-called pure version generates solutions by using the proposed procedure alone. The hybrid approach, on the other hand, combines the pure version with a traditional order-based genetic algorithm. The proposed quantum-inspired order-based evolutionary algorithms have been evaluated for two well-known benchmark applications – the traveling salesman problem (TSP) and the vehicle routing problem (VRP) – as well as in a real problem of line scheduling. Numerical results were obtained for ten cases (7 VRP and 3 TSP) with sizes ranging from 33 to 101 stops and 1 to 10 vehicles, where the proposed quantum-inspired order-based genetic algorithm has outperformed a traditional order-based genetic algorithm in most experiments.

1. Introduction

Ordering problems are extremely broad and relate to many practical applications: allocation of crews in air routes, reverse logistics, allocation of tasks in parallel processing, vehicle routing and production planning (Gopalakrishnan, & Johnson, 2005; Özbakir, & Tapkan, 2011; Sbihi, & Eglese, 2007).

The most well-known of such applications is possibly the travelling salesman problem (TSP) (Applegate, Bixby, Chvatal, & Cook, 2006; Falcone, Chen, & Hamad, 2013), which can be described as follows. Given n locations and assuming that the salesman is in location 1, what should be the sequence of visits, ensuring that each city is visited exactly once, that minimizes the total distance traveled, including the return to location 1? It can be solved through diverse techniques: Branch & Cut (Padberg, & Rinaldi, 1991), Lagrangean Relaxation (Herrero, Ramos, & Guimarans, 2010), Genetic Algorithms (Rani, & Kumar, 2014; Roy, Panja, & Sardar, 2014), Ant colonies (Dorigo, & Gambardella, 1997), Particle Swarm (Kennedy, & Eberhart, 1995), Simulated Annealing (Bayran, & Sahin, 2013), Tabu Search (Malek, Guruswamy, Pandya, & Owens, 1989) and Lin-Kernighan heuristic (Karapetyan, & Gutin, 2011).

Another well-known ordering problem is Vehicle Routing (VRP), where every customer in a certain geographic region has a known demand for products. The demand of each customer must be met in a single delivery. Delivery vehicles start their journey from a warehouse, to which they must return after completing their deliveries. The goal is to determine which vehicle should serve each client so that the total distance traveled by all vehicles is minimized. In addition, the allocation of vehicles to customers must take into account the maximum capacity of each vehicle.

The TSP can be considered a particular case of the VRP with only one vehicle whose capacity is equal to the sum of all customers' demands. Thus, a variety of techniques that solve the VRP can also be applied to the TSP, such as: Branch & Cut (Fukasawa et al., 2006), Heuristics (Pichpibul, & Kawtummachai, 2013) and Evolutionary Computation techniques, such as Genetic Algorithms (Nazif, & Lee, 2012; Prins, 2004), Ant Colonies (Bin, Zhong-Zhen, & Baozhen, 2009), Particle Swarm (Marinakis, & Marinaki, 2010), Simulated Annealing (Breedam, 1995) and Tabu Search (Gendreau, Hertz, & Laporte, 1994).

More recently, a new class of evolutionary computation algorithms, inspired by quantum computing principles, has been developed to achieve better performance in computationally

* Corresponding author. Fax: +55 21 3527 1232.
E-mail addresses: lucianoreisdasilveira@gmail.com (L.R. da Silveira), ricardo@ele.puc-rio.br (R. Tanscheit), marley@ele.puc-rio.br (M.M.B.R. Vellasco).

http://dx.doi.org/10.1016/j.eswa.2016.08.067
0957-4174/

intensive problems, called *quantum-inspired evolutionary algorithms*, (Cruz, Vellasco, & Pacheco, 2008; Dias, & Pacheco, 2012; Han, & Kim, 2000, 2002; Pinho, Vellasco, & Cruz, 2009; Vellasco, Cruz, & Pinho, 2010). In some applications, time taken in the calculation of the evaluation function may be critical. In those cases, reaching good solutions with the smallest possible number of evaluations is a very important factor. These quantum-inspired evolutionary models provide good (or optimal) solutions with a smaller number of evaluations, being adequate to solve optimization problems where the evaluation of each possible solution is computationally expensive. These algorithms have been previously used to solve combinatorial and numerical optimization problems, based on binary (Han, & Kim, 2002; Pinho et al., 2009) and real representations (Cruz et al., 2008; Pinho et al., 2009; Vellasco et al., 2010), respectively, providing better results than classical genetic algorithms, with less computational effort. However, for ordering problems, order-based genetic algorithms are more suitable than those with binary and real representations. This is because specialized crossover and mutation processes are employed to always generate feasible solutions.

This work proposes a new quantum-inspired evolutionary algorithm for solving ordering problems with a reduced number of evaluations. The so-called Quantum-Inspired Order-Based Evolutionary Algorithm (QIEA-O) was developed to solve any ordering problem (see Section 2.1), irrespective of the way the cost (evaluation) function is computed and of the elements in the universe, considering permutation or repetition. Quantum-inspired approaches have already been used in ordering problems. However, as discussed in Section 2.2, those approaches are application specific, making use of quantum bits and rotation matrices to solve specific ordering problems, such as the traveling salesman or the knapsack problem. The representation proposed in this paper makes QIEA-O model application independent, allowing it to be applied to any ordering problem.

According to Mark Moore (Moore, & Narayanan, 1995) and Ajit Narayanan (Narayanan, & Moore, 1996), an algorithm, to be considered as quantum inspired, must have the following characteristics:

1. The problem must have a numeric representation, or a method for converting it into a numerical representation should be employed;
2. An initial configuration must be determined;
3. A stopping criterion must be defined;
4. The problem should be able to be divided into smaller subproblems;
5. The number of universes (or superposition states) must be identified;
6. Each problem must be associated with one of the universes;
7. The calculations in different universes must occur independently;
8. Some form of interaction between multiple universes must exist. This interference should allow finding a solution to the problem or provide information for each sub-problem in each universe to be able to find it.

A note should be made with respect to the concept of the universe mentioned above. In quantum mechanics, a particle (an electron, for example) does not have a particular position until it is effectively measured. It is considered that there is a probability distribution for the possible positions of the electron everywhere in space. In other words, the electron behaves as if it could be in infinite locations at the same time. One interpretation of this phenomenon is that the electron exists in several universes simultaneously, and its position is only determined when a measurement is made and the collapse of the probability occurs (the electron ceases to exist in other universes) (Moore, & Narayanan, 1995).

An alternative definition for an algorithm to be considered quantum inspired is presented in Zhang (2010). According to this author, a QIEA (Quantum Inspired Evolutionary Algorithm) is characterized as an evolutionary algorithm where quantum individuals are represented by quantum bits (Qbits) and quantum gates (Qgate) are employed to update individuals. Under this definition Zhang (2010), the proposed algorithm, as well as others in the literature (Cruz et al., 2008; Pinho et al., 2009; Vellasco et al., 2010), might not be considered quantum inspired, since it does not use bits or quantum gates.

The algorithm presented in this paper is based on quantum computing principles established by Moore (Moore, & Narayanan, 1995) and Ajit Narayanan (Narayanan, & Moore, 1996), which allow the optimization process to be carried out with a smaller number of evaluations without using rotation matrices.

This paper has four additional sections. The next section describes the main concepts of Quantum-Inspired Order-Based Evolutionary Algorithm. Section 3 presents the proposed QIEA-O model in details. Section 4 presents the evaluation of the proposed model in two classic problems (the TSP and the VRP, with sizes ranging from 33 to 101 stops, and 1 to 10 vehicles) and a real problem of line scheduling. Finally, Section 5 concludes the work.

2. Background

2.1. Ordering problems

The mathematical formulation of the ordering problem under consideration in this work is:

Minimize $f(\boldsymbol{x})$
Subject to: $\boldsymbol{x} \in X = \{(x_1, x_2, \ldots, x_n),$ such that $x_i \in$ U, $\forall i \in \{1, 2, \ldots, n\}$ and U finite$\}$
where, f: $X \rightarrow R$
$\boldsymbol{x} \rightarrow f(\boldsymbol{x})$

There are two possibilities about the admissible values for a solution:

(1) The solutions $\boldsymbol{x}$ do not allow repetitions of the elements in set U. This is the ordering permutation problem. In this case, the values of n must be equal to m. The TSP is an example of this kind of problem.
(2) The solutions $\boldsymbol{x}$ do allow repetitions of the elements in set U. This is the ordering repetition problem. Notice that in this case the values of n and m have no relationship to each other, although in the most situations $n \geq m$. The production planning is an example of this kind of problem.

2.2. Quantum inspired evolutionary algorithms

Algorithms based on evolutionary computation generate a sequence of solutions along iterations (called generations) so that the value of a criterion function gradually improves.

In genetic algorithms with binary representation, each gene of an individual is represented by a bit. In the case of quantum-inspired genetic algorithms with binary representation, each gene of an individual is represented by a quantum bit.

One quantum bit (Qbit) is the smallest amount of information stored in a quantum computer. Differently from the bit of conventional computing, the value of the Qbit will not always be the same; this value will depend on its observation. In other words, it is not possible to determine a priori the measured value (state) of a Qbit; it is only possible to estimate the probability of observing value 0 or value 1.

Making use of the Dirac notation, a Qbit is represented by: $|\psi = \alpha|0\rangle + \beta|1\rangle$, where α, $\beta \in$ C, $|\alpha|^2 + |\beta|^2 = 1$, $|\alpha|^2$ and $|\beta|^2$ are

the probabilities of obtaining state 0 and state 1, respectively, when the Qbit is observed.

A 2Qbit is represented by $\delta_0|00\rangle+\delta_1|01\rangle+\delta_2|10\rangle+\delta_3|11\rangle$, where $\delta_i \in C$, $\forall i \in \{0, 1, 2, 3\}$ and $|\delta_0|^2+|\delta_1|^2+|\delta_2|^2+|\delta_3|^2=1$. Thus, a 2Qbit may produce four states (00, 01, 10, 11) upon being observed.

In general, terms, an mQbit is given by:

$\delta_0 \ |00\ldots00\rangle+\delta_1|00\ldots01\rangle+\delta_2|00\ldots10\rangle+\delta_3|00\ldots11\rangle+\ldots..+\delta_{2^m-2}|11\ldots10\rangle+\delta_{2^m-1}|11\ldots11\rangle$, where $\delta_i \in C, \forall i \in \{0, 1, \ldots, 2^m-1\}$, and $\sum_{i=0}^{2^m-1}|\delta_i|^2=1$. In this case, 2^m states may be observed, each of them with a given probability (Rieffel, & Polak, 2000).

The idea behind quantum-inspired genetic algorithms is to generate each classic individual based on the observations of quantum individuals. To obtain a classic gene from a quantum gene, the quantum bit must be observed in a random way. Thereafter, for each generation, the observation process is performed several times on the quantum individuals and a set of solutions associated to each of them is obtained. Some criterion is then adopted to choose one solution for each of those sets. In other words, one solution will be associated to each quantum individual. Finally, the quantum individual is updated, producing a new population. This procedure is repeated and the value of the criterion function is expected to improve.

Many researchers have applied quantum-inspired genetic algorithms, based on QBits and rotation matrices, to different optimization problems, such as: energy dispatching (Babu, Das, & Patvardhan, 2008; Chung, Yu, & Wong, 2011; Vlachogiannis, & Lee, 2008); faces recognition (Jang, Han, Kim, Yu, & Wong, 2004); robotics (Kim, & Kim, 2009; Li, Xu, Yang, Chen, & Li, 2009); data clustering (Xiao, Yan, Lin, Yuan, & Zhang, 2008); metallurgy (Setia, & Hans, 2012); SVM tuning parameters (Luo et al., 2008); school courses timetabling (Chaturvedi, 2013); among many others.

Specifically for order optimization problems, Han and Kim (2000) and Han and Kim (2002) developed quantum inspired algorithms for the knapsack problem and Talbi, Draa, and Batouche (2004) proposed an extension of Han and Kim (2000) and Han and Kim (2002) models to solve the traveling salesman problem. More recently, Ning, Guo, and Chen (2015) proposed a quantum-inspired genetic algorithm for the vehicle routing problem (VRP) and (Wang, Wu, & Zheng, 2005) for scheduling problems. All these algorithms make use of QBits and rotation matrices, with chromosome representations that are tuned for solving specific ordering problems.

In this paper, a new quantum-inspired genetic algorithm for ordering problems is proposed, whose representation of the quantum individual is based on a vector of mQbits, producing a generic chromosome representation that can be applied to a wide range of ordering problems.

The next section describes, in details, the quantum individual representation, the initialization process and the complete evolutionary process of the Quantum-Inspired Order-Based Evolutionary Algorithm (QIEA-O) proposed in this work.

3. Quantum inspired order-based genetic algorithms

The evolutionary process of the Quantum-Inspired Order-Based Evolutionary Algorithm (QIEA-O) is based on a quantum population of quantum individuals, as described in the previous section. For each generation, an observation process is performed several times on the quantum individuals, obtaining a set of solutions from each of them.

Two different observation processes have been developed, depending on the type of ordering problem: Permutation and Repetition. The details of the proposed algorithms are presented in the following sections.

3.1. Quantum individual for ordering problems

Assuming a universe $U=\{1, 2,\ldots.., m\}$, the vector $\boldsymbol{x}=(x_1, x_2, \ldots., x_n)$, where $x_i \in U$, is a solution for an ordering problem (see Section 2.1). Every x_i assumes one of the values of the elements in U with a certain probability. The sum of the probabilities of the elements being observed must be equal to 1.

In order to define a quantum individual for an ordering problem, the components of a vector representing the quantum individual may be represented by a specific mQbit. For example, if U has three elements, only states $|001\rangle$, $|010\rangle$ and $|100\rangle$ of the 3Qbit shall be considered (these are the states that can represent the first, second and third elements of U, respectively). Therefore, these are the only states that must present non-zero probability of being observed; the sum of these probabilities must be 1.

The mQbit that represents each component of the quantum individual vector is of the form:

$$m\text{Qbit} = \tau_1|00\ldots01\rangle+\tau_2|00\rangle\ldots10\rangle+\ \ldots.\ +\ \tau_{m-1}|01\ldots00\rangle + \tau_m|10\ldots00\rangle,$$

where $\tau_i \in C$ e $\sum_{i=1}^{m}|\tau_i|^2=1$.

A simpler notation for the quantum individual, proposed in this work, is the following n x m matrix

$$\mathbf{Q}=\begin{bmatrix} q_{11} & \cdots & q_{1m} \\ \vdots & \ddots & \vdots \\ q_{n1} & \cdots & q_{nm} \end{bmatrix} \in R^{nxm}, q_{ij} \in [0, 1]\ ;$$

$$\sum_{j=1}^{m} q_{ij}=1, \quad \forall i \in \{1, \ldots, n\} \tag{1}$$

where the elements in each row represent the probability of each mQbit state to be observed.

3.2. Initialization of the quantum individual

Any set of values $q_{ij}, i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$ that satisfies (1) may be used to initialize a quantum individual. However, in order to avoid favoring a particular solution, a good approach is to initialize all solutions in the search space with equal probability. This can be implemented as:

$$q_{ij}=\frac{1}{m}, \ \forall\ i \in \{1, \ldots, n\} \text{ e } j \in \{1, \ldots, m\}$$

Consider, for example, a quantum individual with dimension $n=2$ and $m=3$. Then, the quantum individual would be initialized as follows:

$$\begin{vmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{vmatrix}$$

3.3. Observation of the quantum individual

Once a quantum individual is generated, the next action is to generate classic individuals (problem solutions), a process called observation of the quantum individual. This consists of randomly selecting one column for each row of the quantum individual. A quantum individual may be observed several times, so that many different solutions are obtained. Depending on the type of the ordering problem, the solutions vector may consist of permutations of the elements of U or there may be repetitions of those elements in that vector. This is explained below.

	Before Update	After Update
Row 2	$(q_{21}, q_{22}, q_{23}, q_{24})$	$\left(\frac{q_{21}}{(1-q_{23})}, \frac{q_{22}}{(1-q_{23})}, 0, \frac{q_{24}}{(1-q_{23})}\right) = (q'_{21}, q'_{22}, 0, q'_{24})$
Row 3	$(q_{31}, q_{32}, q_{33}, q_{34})$	$\left(\frac{q_{31}}{(1-q_{33})}, \frac{q_{32}}{(1-q_{33})}, 0, \frac{q_{34}}{(1-q_{33})}\right) = (q'_{31}, q'_{32}, 0, q'_{34})$
Row 4	$(q_{41}, q_{42}, q_{43}, q_{44})$	$\left(\frac{q_{41}}{(1-q_{34})}, \frac{q_{42}}{(1-q_{34})}, 0, \frac{q_{44}}{(1-q_{34})}\right) = (q'_{41}, q'_{42}, 0, q'_{44})$

Fig. 1. Updating rows 2, 3 and 4 of quantum individual.

	Before Update	After Update
Row 3	$(q'_{31}, q'_{32}, 0, q'_{34})$	$\left(\frac{q'_{31}}{(1-q'_{32})}, 0, 0, \frac{q'_{34}}{(1-q'_{32})}\right) = (q''_{31}, 0, 0, q''_{34})$
Row 4	$(q'_{41}, q'_{42}, 0, q'_{44})$	$\left(\frac{q'_{41}}{(1-q'_{42})}, 0, 0, \frac{q'_{44}}{(1-q'_{42})}\right) = (q''_{41}, 0, 0, q''_{44})$

Fig. 2. Updating rows 3 and 4 of quantum individual.

3.3.1. Observation for the permutation problem

Consider that the first row of the matrix that represents the quantum individual is expressed by $(q_{11}, q_{12}, ..., q_{1m})$. Upon selection of a random number $r_1 \in (0,1]$, the kth element of U will be chosen if $\sum_{j=1}^{k} q_{1j} \leq r_1$ and $\sum_{j=1}^{k+1} q_{1j} > r_1$. Assuming that the p-th element has been chosen when row 1 is processed (x_1 will assume the value in U associated to the element p), the next step consists of determining x_2. As no repetition is allowed, p is no longer eligible. Thus, the probability of again selecting p must be annulled for any x_i, $i > 1$, i.e. elements $q_{ip}\ \forall\ i \in \{2, \ldots, n\}$ must be zeroed. As this may not satisfy $\sum_{j=1}^{m} q_{ij} = 1, i \in \{2, \ldots, n\}$ any more, all elements from the second row onwards must be updated. If the elements are updated by the expression

$$q'_{ij} = \frac{q_{1j}}{1 - q_{ip}},\ \forall\ i \in \{2, \ldots, n\},\ \ j \in \{1, \ldots, m\},\ \ j \neq p \text{ and}$$
$$q'_{ip} = 0, \forall\ i \in \{2, \ldots, n\}, \tag{2}$$

$\sum_{j=1}^{m} q'_{ij} = 1, i \in \{2, \ldots, n\}$ will remain valid for all q'_{ij}, $\forall\ i \in \{2, \ldots, n\},\ j \in \{1, \ldots, m\}$. Another element than p will then be selected. This process is repeated for the determination of the other elements of the solutions vector. This needs not to be done in a natural order; anyone is acceptable and will be represented by vector $o = (o_1, o_2, \ldots., o_n)$.

Consider an example of a 4×4 problem where the solution (3,2,1,4) is obtained after the observations process. When the first row is processed, the column 3 is selected and the rows 2, 3 and 4 must be updated according to Fig. 1.

Next, column 2 is chosen when row 2 is processed, resulting in a new updating of rows 3 and 4 to avoid column 2 to be chosen again. Fig. 2 presents the updating process of rows 3 and 4.

Finally, row 3 selects column 1 and Fig. 3 exhibits the updating process of row 4. Notice that when row 4 is processed, the only remaining choice is column 4.

In addition, the probability of observing a solution $v = (v_1, v_2, \ldots., v_n)$ is given by (Silveira, Tanscheit, & Vellasco, 2012; Silveira, 2014):

$$\prod_{k=1}^{n} \frac{q_{o_k v_{o_k}}}{\sum_{j=k}^{m} q_{o_k v_{o_j}}} \tag{3}$$

Given an order of observation, the sum of the probabilities of all solutions (i.e. all permutations) is equal to 1.

3.3.2. Observation for the repetition problem

Differently from the permutation case, in this case the p-th column may be chosen again. Therefore, the k-th element may be chosen if $\sum_{j=1}^{k} q_{2j} \leq r_2$ and $\sum_{j=1}^{k+1} q_{2j} > r_2$, where r_2 is a random number. Thus, the observation process consists of obtaining a sequence of random numbers r_i, $i \in \{1, \ldots, n\}$ to determine which column will be chosen for each row of the solutions vector. The probability of the j-th element of U being allocated to the i-th component of the solutions vector is equal to q_{ij}. The probability of observing a solution $v = (v_1, v_2, \ldots., v_n)$ will be $\prod_{i=1}^{n} q_{iv_i}$. As in the permutation problem, the sum of the probabilities of all possible solutions is equal to 1.

In both cases, permutation and repetition observation methods, many solutions can be generated from a single quantum individual. Therefore, a criterion must be defined to select, among the various solutions generated, one that will be used in the next step, which consists of updating the quantum individual. The possibility of applying a mutation to the chosen solution may be considered. The probability of the application of a mutation is controlled by a parameter called $TMut \in [0, 1]$. If a randomly generated number in the interval [0, 1] is smaller than $TMut$, mutation will be applied. This consists of swapping two randomly selected positions in the solution vector.

3.4. Updating the quantum individual

Once the observation and mutation stages have been completed, to each quantum individual there is a corresponding

	Before Update	After Update
Row 4	$(q''_{41}, 00, q''_{44})$	$\left(0,0,0,\frac{q''_{44}}{(1-q''_{41})}\right) = (0,0,0,q'''_{44})$

Fig. 3. Updating row 4 of quantum individual.

$$\begin{vmatrix} q^t_{11} & q^t_{12} & q^t_{13} & q^t_{14} \\ q^t_{21} & q^t_{22} & q^t_{23} & q^t_{24} \\ q^t_{31} & q^t_{32} & q^t_{33} & q^t_{34} \\ q^t_{41} & q^t_{42} & q^t_{43} & q^t_{44} \end{vmatrix} = (1-\varepsilon)\begin{vmatrix} q^{t-1}_{11} & q^{t-1}_{12} & q^{t-1}_{13} & q^{t-1}_{14} \\ q^{t-1}_{21} & q^{t-1}_{22} & q^{t-1}_{23} & q^{t-1}_{24} \\ q^{t-1}_{31} & q^{t-1}_{32} & q^{t-1}_{33} & q^{t-1}_{34} \\ q^{t-1}_{41} & q^{t-1}_{42} & q^{t-1}_{43} & q^{t-1}_{44} \end{vmatrix} + \varepsilon\begin{vmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} =$$

$$\begin{vmatrix} (1-\varepsilon)q^{t-1}_{11} & (1-\varepsilon)q^{t-1}_{12} & (1-\varepsilon)q^{t-1}_{13}+\varepsilon & (1-\varepsilon)q^{t-1}_{14} \\ (1-\varepsilon)q^{t-1}_{21} & (1-\varepsilon)q^{t-1}_{22}+\varepsilon & (1-\varepsilon)q^{t-1}_{23} & (1-\varepsilon)q^{t-1}_{24} \\ (1-\varepsilon)q^{t-1}_{31}+\varepsilon & (1-\varepsilon)q^{t-1}_{32} & (1-\varepsilon)q^{t-1}_{33} & (1-\varepsilon)q^{t-1}_{34} \\ (1-\varepsilon)q^{t-1}_{41} & (1-\varepsilon)q^{t-1}_{42} & (1-\varepsilon)q^{t-1}_{43} & (1-\varepsilon)q^{t-1}_{44}+\varepsilon \end{vmatrix}$$

Fig. 4. Updating a quantum individual 4×4 using the solution (3,2,1,4). Where the upper index, in each entry of the matrix, indicates the generation.

solution. Those individuals must now be altered so that the next observation generates better solutions.

Assuming that upon the first observation process (1st generation) a solution $\boldsymbol{e}(1) = \{e_1(1),\ e_2(1), \ldots,\ e_n(1)\}$ for the ordering problem has been chosen (the best solution in terms of the evaluation function, for instance) for the initial quantum individual $\mathbf{Q}(0)$, this must be updated so that the probability of observing $\boldsymbol{e}(1)$ increases at the next observation (2nd generation). This can be guaranteed by the following expression:

$$\mathbf{Q}(1) = (1-\varepsilon)\mathbf{Q}(0) + \varepsilon\mathbf{E}(1) \tag{4}$$

where $\mathbf{Q}(1)$ is the updated quantum individual, $\varepsilon \in [0, 1]$ is a parameter to be specified and $\mathbf{E}(1)$ is an n x m matrix such that their elements $E_{ij}(1)$ are equal to 1 if $j = e_j(1)$ and 0 otherwise. That is, $\mathbf{E}(1)$ is an identity matrix the rows of which are ordered in accordance with $\boldsymbol{e}(1)$. Generalizing this for a generation t, the updating expression will be:

$$\begin{aligned} &\mathbf{Q}(t) = (1-\varepsilon)\mathbf{Q}(t-1) + \varepsilon\mathbf{E}(t) \text{ or} \\ &q_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) + \varepsilon E_{ij}(t) \end{aligned} \tag{5}$$

where $E_{ij}(t) = 1$ if $j = e_j(t)$ and 0 otherwise; $\boldsymbol{e}(t)$ is the solution at the end of generation t. Fig. 4 presents an example of a 4×4 quantum individual updated using the solution (3,2,1,4).

It must be noted that:

(a) The conditions expressed in (1) are always satisfied:
  (a.1) $q_{ij} \in [0, 1]$, $i \in \{1, \ldots, n\}$ e $j \in \{1, \ldots, m\}$:
  If $e_{ij}(t)=0$ then $q_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) + \varepsilon e_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) \in [0, 1]$;
  If $e_{ij}(t)=1$ then $q_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) + \varepsilon e_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) + \varepsilon \in [0, 1]$ (since $q_{ij}(t)$ is a convex combination of values between 0 and 1).
  (a.2) $\sum_{j=1}^{m} q_{ij}(t) = 1,\ \forall i \in \{1, \ldots, n\}$:

$$\begin{aligned} \sum_{j=1}^{m} q_{ij}(t) &= \sum_{j=1}^{m}(1-\varepsilon)q_{ij}(t-1) + \sum_{j=1}^{m}\varepsilon e_{ij}(t) \\ &= (1-\varepsilon)\sum_{j=1}^{m} q_{ij}(t-1) + \varepsilon\sum_{j=1}^{m} e_{ij}(t) \\ &= (1-\varepsilon) + \varepsilon = 1,\ \forall i \in \{1, \ldots, n\}, \end{aligned}$$

(b) The probability of observing $\boldsymbol{e}(t)$ with $\mathbf{Q}(t)$ is equal or higher than for $\mathbf{Q}(t-1)$:
  If $e_{ij}(t) = 0$ then $q_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) + \varepsilon e_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) \le q_{ij}(t-1)$;
  If $e_{ij}(t) = 1$ then $q_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) + \varepsilon e_{ij}(t) = (1-\varepsilon)q_{ij}(t-1) + \alpha \ge q_{ij}(t-1)$.

Regarding parameter ε, as it increases, the chance of observing $\boldsymbol{e}(t)$ from $\mathbf{Q}(t)$ also increases (with respect to the chance for $\mathbf{Q}(t-1)$). If $\varepsilon=1$, $\mathbf{Q}(t) = \mathbf{E}(t)$: the quantum individual is represented by $\boldsymbol{E}(t)$ and the probability of observing $\boldsymbol{e}(t)$ will be 1.

In the updating process, a particular same solution may always be produced (as in the case above, for $\varepsilon=1$). This is undesirable since few new solutions will be obtained in the observation process. In situations like this, the quantum individual is considered to have reached saturation. That is, for a saturated individual there is a high probability of always observing the same solution; in other words, the probability of observing different solutions is very low.

It should be noted that, when composing the quantum individual, the weight of each solution decreases exponentially along generations. Considering Eq. (5) for $t = N$:

$$\begin{aligned} \mathbf{Q}(N) &= (1-\varepsilon)\mathbf{Q}(N-1) + \varepsilon\mathbf{E}(N) = (1-\varepsilon)^N\mathbf{Q}(0) \\ &+ \sum_{j=1}^{N-1}((1-\varepsilon)^{N-j}\varepsilon\mathbf{E}(j) + \varepsilon\mathbf{E}(N) \end{aligned} \tag{6}$$

It can be seen that the role played by each solution $\mathbf{E}(t)$ in the formation of the quantum individual $\mathbf{Q}(N)$ increases with time, i.e. more recent solutions have a higher weight in $\mathbf{Q}(N)$.

3.5. Dealing with the saturated quantum individual

Assume that q_{ij} are the elements of the quantum individual matrix and that $s_i = \mathrm{Max}_{j=1,\ldots,m}(q_{ij}),\ i \in \{1, \ldots, n\}$ is the element with the largest value in row i. The measure of saturation of the quantum individual will be given by $s = \mathrm{Min}_{i=1,\ldots,n}(s_i)$; the individual will be considered as saturated when s is larger than a given value (0.999 or $0.999^{1/n}$, for example). If identified as saturated, the quantum individual will no longer be observed or updated; the best solution obtained throughout the generations will be recovered and used as the one produced by this individual.

3.6. Quantum inspired order-based genetic algorithm: summary

The proposed quantum-inspired algorithm, with its initialization, observation and updating procedures, presented in the previous sections, has been applied in two different approaches. In the first one, called "pure", only the solutions obtained through the observation of quantum individuals will be considered. In the second approach, called "hybrid", crossover and mutation will be additionally employed over the classic individuals, as an attempt to improve solutions even further.

The following pseudocode summarizes the steps described in the previous subsections and considers the more general, or hybrid, approach.

```
Begin
  Read parameters
  Begin quantum population
  t ← 0
  while (t ≤ NGer)
    iqua ← 0
    /* observation process */
    while ((iqua ≤ NQua) and (iqua is not saturated))
      iqua ← iqua+1
      iobs ← 0
        while (iobs ≤ NObs)
          iobs ← iobs+1
          sol ← Observe Quantum Individual (iqua)
          r ← random number between 0 and 1
          if (r < TMut) then sol ← do mutation (sol)
          Evaluate solution (sol) and store results
        end while
    end while
    /* updating process */
    while ((iqua ≤ NQua) and (iqua is not saturated))
      iqua ← iqua+1
      solesc ← Choose solution to update quantum individual (iqua)
      ε ← Choose parameter to update quantum individual (iqua)
      Update quantum individual (iqua) with solution solesc and parameter ε
      Saturation index ← Calculate Saturation Index (iqua)
      if (Saturation Index < Lim) then quantum individual not saturated
        ifnot quantum individual saturated
    end while
    if (t ∈ CGera) then
      /* create initial population */
      iqua ← 0
      while (iqua ≤ NQua)
        iqua ← iqua+1
        Observe NCla times the quantum individual iqua
        Store the observed solutions in the initial population
      End while
      Process the ordering genetic algorithm with the initial population
obtained from observations of quantum individuals and parameters (NGerc, TCruc,
TMutc, FElic)
      while if
    end while
End
```

All the necessary parameters for the execution of the algorithm are shown in Table 1.

Table 1
Parameters definition of the proposed quantum-inspired order-based genetic algorithm.

Parameter	Description
NGer	Maximum number of generations.
NQua	Number of quantum individuals in the quantum population.
NObs	Number of observations (classical individuals) from each quantum individual, at each generation.
TMut	Mutation rate applied to the best-observed solution.
ε	Value for updating the quantum individual (see Eq. 4).
Lim	Limiting value above which the quantum individual is considered saturated.
The following six parameters are exclusive for the "hybrid" approach (**CGera** $\neq \emptyset$)	
CGera	Set of generations when the classical order-based genetic algorithm is executed.
NCla	Number of observations from each quantum individual for the initial population of the classical order-based genetic algorithm.
NGerc	Maximum number of generation for each execution of the classical order-based genetic algorithm.
TCroc	Crossover Rate for the classical order-based genetic algorithm.
TMutc	Mutation Rate for the classical order-based genetic algorithm.
FElic	Elitism for the classical order-based genetic algorithm

4. Case studies

The proposed Quantum-Inspired Order-Based Genetic Algorithms (pure and hybrid) have been applied to two different problems: i) Permutation, where 3 instances of the TSP and 7 of the VRP are evaluated; ii) and Repetition, with an actual case of a line scheduling problem. The following subsections present the comparative results obtained by both "pure" and "hybrid" algorithms with a traditional order-based genetic algorithm.

4.1. Case study: permutation problem

The permutation problem falls in the category of ordering combinatorial optimization problems, which can be formulated as:

Minimize $f(\boldsymbol{x})$
Subject to: $\boldsymbol{x} \in X = \{(x_1, x_2, \ldots, x_n)$, such that $x_i \in U, \forall i \in \{1, 2, \ldots, n\}$ and U finite$\}$
where f: $X \rightarrow$ R
$\boldsymbol{x} \rightarrow f(\boldsymbol{x})$

Perhaps the most well-known permutation problem is the TSP. If U is the set of the n locations to be visited, then X will be formed by all possible permutations of $n-1$ locations (the initial one is always location 1). The criterion, or cost, function will be the minimum distance travelled starting from location 1 and visiting only once each of the n-1 locations before returning to location 1. If, for example, $U = \{1, 2, 3, 4\}$ represents the four locations to be visited, then X will be expressed by: {(1,2,3,4), (1,2,4,3), (1,3,2,4), (1,3,4,2), (1,4,2,3), (1,4,3,2)}. The criterion function for an element (1, i, j, k) of X will be given by: $f(1,i,j,k) = D(1,i) + D(i,j) + D(j,k) + D(k,1)$, where $D(a,b)$ is the distance travelled between locations a and b.

Another well-known permutation problem is the VRP. The solution vectors $\boldsymbol{x}$ are permutations of the elements of U and the criterion function expresses the total distance travelled by all vehicles, starting from and returning to the same initial point (a warehouse, for example), and penalizes violations of vehicles capacities.

Considering n stopping points, each of them requiring an amount d(i) $\in$ N, $i \in$ {1,....,n}, and v vehicles, with capacities cap(j) $\in$ N, $j \in$ {1,....,v}, a possible solution is a vector with $v+n$ coordinates, with v coordinates equal to zero and n coordinates numbered from 1 to n (Bjarnadóttir, 2004; Wu, Ji, & Wang, 2008). Each zero coordinate specifies the beginning of a specific route. It must be ensured that $\sum_{i=1}^{n} d(i) \leq \sum_{j=1}^{v} cap(j)$ for a solution to be viable. As an example, consider a case with 9 stopping points ($n = 9$) and 3 vehicles ($v = 3$), each with a different capacity: 4, 3 and 2, respectively. A possible solution for this case would be (0,1,2,3,7,0,4,5,6, 0,8,9), where vehicle 1 stops at points 1,2,3 and 7; vehicle 2 at points 4,5 and 6; and vehicle 3 stops at points 8 and 9.

Seven instances of the VRP and three of the TSP have been obtained from http://www.branchandcut.org and http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ respectively. The TSPs are att48.tsp, berlim52.tsp and kroc100.tsp (with 48, 52 and 100 locations). The VRPs are E-n33-k4.vrp, B-n41-k6.vrp, F-n45-k4.vrp, P-n50-k10.vrp, A-n60-k9.vrp, B-n80-k10.vrp e E-n101-k8.vrp. For the first one, for example, there are 32 stopping points plus the initial one and 4 vehicles, all of them with the same capacity. The same

Table 2
Instances, optimal solutions and number of evaluations.

Instance	Opt. Sol.	Low level			Medium level			High level		
		NGer	NPop	NEval	NGer	NPop	NEval	NGer	NPop	NEval
E-n33-k4	835	1650	42	69.342	3300	83	273.983	3300	330	1.089.330
B-n41-k6	829	2050	52	106.652	4100	103	422.403	4100	410	1.681.410
F-n45-k4	724	2250	57	128.307	4500	113	508.613	4500	450	2.025.450
T-n48-k1	10,628	2400	60	144.060	4800	120	576.120	4800	480	2.304.480
P-n50-k10	696	2500	63	157.563	5000	125	625.125	5000	500	2.500.500
T-n52-k1	7542	2600	65	169.065	5200	130	676.130	5200	520	2.704.520
A-n60-k9	1354	3000	75	225.075	6000	150	900.150	6000	600	3.600.600
A-n80-k10	1763	4000	100	400.100	8000	200	1.600.200	8000	800	6.400.800
T-n100-k1	20,749	5000	125	625.125	10,000	250	2.500.250	10,000	1000	10.001.000
E-n101-k8	815	5000	125	625.125	10,100	251	2.535.351	10,100	1010	10.201.010

rationale applies to the other instances. As the TSP is a particular case of the VRP, instances att48.tsp, berlim52.tsp and kroc100.tsp correspond to T-n48-k1.vrp, T-n52-k1.vrp e T-n100-k1.vrp, where the initial letter T is employed to identify a TSP.

Every problem has been executed at three levels of computational effort (low, medium and high), defined here by the number of evaluations performed. At the high level, the number of evaluations was defined as the number of evaluations the traditional order-based genetic algorithm would perform with a number of generations equal to 100 times the number of stopping points (plus the initial one) and a population size 10 times the number of stopping points (also plus the initial one). In the case of E-n33-k4, for example, this would give $330*(3300+1)=1.089.330$ evaluations (the term $+1$ refers to the computation of the criterion function for the initial population). At the medium level, population was reduced to ¼ of that at the high level. At the low level, population was reduced by half of that at the medium level.

Optimal solutions (Opt.Sol.) and number of evaluations at each level (NEval) are shown in Table 2 for the instances referred to above. Also shown are Nger – number of generations and Npop – population size.

Each instance was executed 10 times for different seeds, obtained through the Mersenne-Twister generator (Matsutomo, & Nishimura, 1998):

Experiment	Seed	Experiment	Seed
1	104.677	6	59.921
2	99.984	7	49.991
3	89.977	8	39.979
4	79.943	9	29.927
5	69.931	10	19.993

For every set of 10 experiments, the best, average, standard deviation and worst solutions have been stored, as well as the number of evaluations.

The traditional order-based genetic algorithm used for comparison has been implemented using roulette for individuals selection, uniform crossover (Gen, & Cheng, 1997) and 10% elitism.

4.1.1. Comparative results

This section presents the comparison of the results obtained by the proposed algorithms ("pure" and "hybrid" versions) with the traditional order-based genetic algorithm.

The set of parameters for each assessed algorithm (proposed algorithms and traditional genetic algorithm) have been determined by minimizing the average cost of the solutions obtained over 10 experiments, for each computational effort. For the traditional order-based genetic algorithm, 61 distinct sets of parameters were considered. As for versions "pure" and "hybrid" of the proposed algorithm, 110 distinct sets of parameters were evaluated. The best parameters for the order-based genetic algorithm are shown in Table 3, where NGer is the number of generations, NPop is the size of population, TCro, TMut are the crossover and mutation rates, respectively, and FEli is the elitism factor.

Table 3
Best set of parameters for the order-based genetic algorithm.

Problem	Comp. effort	NGer	NPop	TCro	TMut	FEli
E-n33-k4	Low	1650	42	0,50	0,75	0.10
	Medium	3300	83	0.55	0.55	0.10
	High	3300	330	1.00	1.00	0.10
E-n41-k6	Low	2050	52	0.80	1.00	0.10
	Medium	4100	103	0.40	0.80	0.10
	High	4100	410	0.10	0.85	0.10
F-n45-k4	Low	2250	57	0.55	0.25	0.10
	Medium	4500	113	0.55	0.10	0.10
	High	4500	450	0.75	0.30	0.10
T-n48-k1	Low	2400	60	0.50	0.30	0.10
	Medium	4800	120	0.60	0.30	0.10
	High	4800	480	0.80	0.00	0.10
P-n50-k10	Low	2500	63	0.90	0.90	0.10
	Medium	5000	125	0.55	0.50	0.10
	High	5000	500	0.40	1.00	0.10
T-n52-k1	Low	2600	65	0.35	0.25	0.10
	Medium	5200	130	0.35	0.15	0.10
	High	5200	520	1.00	0.60	0.10
A-n60-k9	Low	3000	75	0.40	0.45	0.10
	Medium	6000	150	0.20	0.20	0.10
	High	6000	600	0.80	0.10	0.10
A-n80-k10	Low	4000	100	0.40	0.20	0.10
	Medium	8000	200	0.20	0.40	0.10
	High	8000	800	0.30	0.45	0.10
T-n100-k1	Low	5000	125	0.35	0.10	0.10
	Medium	10,000	250	0.40	0.20	0.10
	High	10,000	1000	0.10	0.15	0.10
E-n101-k8	Low	5000	125	0.65	0.40	0.10
	Medium	10,100	251	0.70	0.15	0.10
	High	10,100	1010	0.85	0.35	0.10

Table 4 presents the best parameters for "pure" version of the proposed algorithm and the parameters are those presented in Table 1, with the value of parameter Lim set equal 0.99.

Table 5 exhibits the best parameters for "hybrid" version of the proposed algorithm. The parameters are those presented in Table 1. The following parameters were set to fixed values: CGera = {NGer}, TMut = 0, $\varepsilon = 0.02$, FElic = 0.10.

In order to improve the comparison among the three methods, the best average value obtained with each algorithm is evaluated. Using the best set of parameters obtained with 10 experiments (Tables 3–5 above), 30 experiments were performed for each problem and computational effort. Table 6 presents the obtained results. A hypothesis test is performed to compare the results. The null hypothesis verifies if the average of the proposed algorithm (both versions) is better than the average of the tradi-

Table 4
Best set of parameters for "pure" version of proposed algorithm.

Problem	Comp. effort	NGer	NQua	NObs	TMut	ε
E-n33-k4	Low	3300	1	20	0,80	0.04
	Medium	8250	1	33	0.70	0.02
	High	8250	4	33	0.90	0.02
E-n41-k6	Low	4100	1	25	0.80	0.14
	Medium	10,250	1	41	1.00	0.04
	High	10,250	4	41	0.50	0.02
F-n45-k4	Low	4500	1	28	0.10	0.20
	Medium	11,250	1	45	0.90	0.02
	High	11,250	4	45	0.30	0.02
T-n48-k1	Low	4800	1	30	0.90	0.20
	Medium	12,000	1	48	0.90	0.02
	High	12,000	4	48	1.00	0.02
P-n50-k10	Low	5000	1	31	0.90	0.12
	Medium	12,500	1	50	1.00	0.02
	High	12,500	4	50	0.30	0.02
T-n52-k1	Low	5200	1	32	0.5	0.02
	Medium	13,000	1	52	0.5	0.02
	High	13,000	4	52	0.6	0.02
A-n60-k9	Low	6000	1	37	1.00	0.16
	Medium	15,000	1	60	1.00	0.10
	High	15,000	4	60	0.40	0.02
A-n80-k10	Low	8000	1	50	1.00	0.20
	Medium	20,000	1	80	1.00	0.16
	High	20,000	4	80	1.00	0.20
T-n100-k1	Low	10,000	1	62	0.80	0.04
	Medium	25,000	1	100	0.90	0.02
	High	Results not obtained due to processing time				
E-n101-k8	Low	10,100	1	63	1.00	0.12
	Medium	25,250	1	101	1.00	0.10
	High	Results not obtained due to processing time				

tional order-based genetic algorithm. In other words, for "pure" version, H0: $Avg_{Pure} \leq Avg_{Gen}$ and, for the "hybrid version", H0: $Avg_{Hyb} \leq Avg_{Gen}$. As 30 experiments were realized, the t-student threshold $((Avg_{Pure} - Avg_{Gen})/Std_{Pure}$ or $(Avg_{Hyb}\text{-}Avg_{Gen})/Std_{Hyb})$ is 1.699 for 95% of confidence and 29degrees of freedom.

As can be verified from Table 6, the hypothesis H0 for "pure" version is accepted in 10 of 30 pairs Problem/Computational Effort. The results for the "hybrid" version are much better, since the hypothesis H0 is accepted in 30 out of 30 pairs.

A final comparison is performed involving Talbi's algorithm (Talbi et al., 2004) and the "pure" and "hybrid" versions of the propose algorithm. Table 7 presents the results of the comparative.

It must be emphasized that the number of total evaluations for the proposed algorithms, both for "pure" and "hybrid" versions, is always smaller than the number of evaluations accomplished by Talbi's algorithm. As the Talbi's algorithm performs 16 evaluations at each generation, the alternatives for the number of quantum individuals and the total number of observations at each generations is 1–16 (one quantum individual and 16 observations over each quantum individual), 2-8, 4-4 and 8-2. Tables 8 and 9 present the parameters used in the "pure" and "hybrid" versions of the propose algorithm, respectively (see Table 1 for the meaning of the columns). The minimum solution value was considered to select the following parameters. For "pure" version, the parameter Lim is set equal to zero and for "hybrid" versions the following parameters were fixed: CGera = {NGer}, TMut = 0, $\varepsilon = 0.02$ and FElic = 0.10.

4.2. Case study: repetition problem

In this problem, components x_i can assume any value in U. In other words, all elements in X are valid solutions. The problem analyzed here is a simplification of an actual line scheduling case,

Table 5
Best set of parameters for "hybrid" version of proposed algorithm.

Problem	Comp. effort	NGer	NQua	NObs	NGerc	NCla	TCroc	TMutc
E-n33-k4	Low	2062	1	10	1443	33	0,70	0.40
	Medium	8250	1	10	5775	33	0.90	0.60
	High	8250	4	10	5775	33	0.70	0.60
E-n41-k6	Low	2562	1	10	1793	41	0.60	0.80
	Medium	10,250	1	10	7175	41	0.40	0.80
	High	10,250	4	10	7175	41	0.60	0.90
F-n45-k4	Low	2812	1	10	1968	45	0.30	0.10
	Medium	11,250	1	10	7875	45	1.00	0.50
	High	11,250	4	10	7875	45	0.30	0.60
T-n48-k1	Low	3000	1	10	2100	48	0.20	0.20
	Medium	12,000	1	10	8400	48	0.30	0.10
	High	12,000	4	10	8400	48	1.00	0.30
P-n50-k10	Low	3124	1	10	2186	50	0.50	0.70
	Medium	12,500	1	10	8750	50	0.90	0.70
	High	12,500	4	10	8750	50	0.70	1.00
T-n52-k1	Low	3250	1	10	2275	52	0.50	0.20
	Medium	13,000	1	10	9100	52	0.90	0.30
	High	13,000	4	10	9100	52	1.00	0.00
A-n60-k9	Low	3750	1	10	2625	60	0.70	0.70
	Medium	15,000	1	10	10,500	60	0.70	0.80
	High	15,000	4	10	10,500	60	0.20	0.30
A-n80-k10	Low	5000	1	10	3500	80	0.50	0.60
	Medium	20,000	1	10	14,000	80	0.40	0.20
	High	20,000	4	10	14,000	80	0.70	0.80
T-n100-k1	Low	6250	1	10	4375	100	0.20	0.10
	Medium	25,000	1	10	17,500	100	0.40	0.10
	High	25,000	4	10	17,500	100	0.60	0.30
AvgGemE-n101-k8	Low	6312	1	10	4418	101	0.80	0.10
	Medium	25,250	1	10	17,675	101	0.80	0.20
	High	25,250	4	10	17,675	101	0.90	0.10

Table 6
Comparison of the best average values obtained by the proposed algorithm ("pure" and "hybrid") and by the traditional order-based genetic algorithm.

Problem	Comp. effort	Algorithm	Average	Standard deviation	Number evaluations	*t*-test	Action
E-n33-k4	Low	Genetic	912.57	29.41506	2080,260		
		Pure	965.87	79.38754	1968,370	0.6714	Accept H0
		Hybrid	910.77	31.27159	2048,490	−0.0576	Accept H0
	Medium	Genetic	897.50	30.15819	8219,490		
		Pure	894.70	21.91978	6117,273	−0.1277	Accept H0
		Hybrid	881.87	18.37522	7855,650	−0.8508	Accept H0
	High	Genetic	864.37	18.94639	32,679,900		
		Pure	881.27	14.20313	32,268,708	1.1899	Accept H0
		Hybrid	859.60	13.5021	31,499,790	−0.3530	Accept H0
B-n41-k6	Low	Genetic	991.33	70.40234	3199,560		
		Pure	1.069.40	83.2981	1124,730	0.9372	Accept H0
		Hybrid	988.53	68.69096	2975,550	−0.0408	Accept H0
	Medium	Genetic	939.70	61.28086	12,672,090		
		Pure	966.63	56.79113	10,710,460	0.4743	Accept H0
		Hybrid	931.13	61.65805	11,901,810	−0.1389	Accept H0
	High	Genetic	894.03	42.57893	50,442,300		
		Pure	934.13	38.83274	45,901,752	1.0326	Accept H0
		Hybrid	916.07	58.21222	47,607,240	0.3785	Accept H0
F-n45-k4	Low	Genetic	828.20	44.13646	3849,210		
		Pure	1.376.70	110.0282	145,546	3.6453	Reject H0
		Hybrid	830.37	44.87128	2975,550	0.5400	Accept H0
	Medium	Genetic	781.27	35.57896	15,258,390		
		Pure	888.17	31.53948	15,025,215	2.3545	Reject H0
		Hybrid	787.33	35.26503	11,901,810	−0.0258	Accept H0
	High	Genetic	778.37	47.96838	17,283,600		
		Pure	858.20	54.50468	4223,982	2.3090	Reject H0
		Hybrid	751.67	24.34246	13,230,640	0.4141	Accept H0
T-n48-k1	Low	Genetic	11.560.27	321.3343	4321,800		
		Pure	16.274.33	1293.193	322,800	3.6453	Reject H0
		Hybrid	11.781.17	409.0749	3925,770	0.5400	Accept H0
	Medium	Genetic	11.161.50	238.7086	17,283,600		
		Pure	13.551.90	1015.231	4223,982	2.3545	Reject H0
		Hybrid	11.155.13	247.2762	13,230,640	−0.0258	Accept H0
	High	Genetic	11.008.23	141.7203	69,134,400		
		Pure	12.637.97	705.8074	23,341,176	2.3090	Reject H0
		Hybrid	11.078.03	168.5322	53,285,570	0.4141	Accept H0
P-n50-k10	Low	Genetic	814.30	33.45659	4726,890		
		Pure	1.283.60	217.9565	4402,247	2.1532	Reject H0
		Hybrid	807.03	36.84425	4218,030	−0.1972	Accept H0
	Medium	Genetic	781.47	25.92648	18,753,750		
		Pure	806.17	89.01501	16,280,930	0.2775	Accept H0
		Hybrid	799.17	43.64714	16,876,830	0.4055	Accept H0
	High	Genetic	739.57	20.64733	75,015,000		
		Pure	771.07	25.69168	46,046,670	1.2261	Accept H0
		Hybrid	765.60	29.70253	67,507,320	0.8765	Accept H0
T-n52-k1	Low	Genetic	8.391.63	201.1905	5071,950		
		Pure	9.423.57	511.3213	2546,430	2.0182	Reject H0
		Hybrid	8.616.80	320.0083	4510,460	0.7036	Accept H0
	Medium	Genetic	8.145.13	207.0196	20,283,900		
		Pure	9.503.43	588.3842	3575,758	2.3085	Reject H0
		Hybrid	8.167.47	255.503	15,218,480	0.0874	Accept H0
	High	Genetic	7.985.50	190.1599	81,135,600		
		Pure	8.980.23	440.3315	13,681,476	2.2591	Reject H0
		Hybrid	8.138.17	261.5394	61,299,870	0.5837	Accept H0
A-n60-k9	Low	Genetic	1.544.33	56.87706	6752,250		
		Pure	1.681.47	67.59671	1333,880	2.0287	Reject H0
		Hybrid	1.591.77	53.72441	5852,130	0.8829	Accept H0
	Medium	Genetic	1.542.13	51.96601	27,004,500		
		Pure	1.672.33	60.41542	5779,770	2.1551	Reject H0
		Hybrid	1.514.73	33.4982	23,402,130	−0.8180	Accept H0
	High	Genetic	1.516.23	51.04671	108,018,000		
		Pure	1.483.70	38.91285	95,016,720	−0.8361	Accept H0
		Hybrid	1.510.10	37.56892	93,608,520	−0.1633	Accept H0

(*continued on next page*)

Table 6 (*continued*)

Problem	Comp. effort	Algorithm	Average	Standard deviation	Number evaluations	*t*-test	Action
A-n80-k9	Low	Genetic	2.081.23	61.57255	12,003,000		
		Pure	2.346.40	86.33524	1571,180	3.0714	Reject H0
		Hybrid	2.101.43	53.02935	9902,730	0.3809	Accept H0
	Medium	Genetic	2.019.53	59.13867	48,006,000		
		Pure	2.299.57	75.75869	5656,190	3.6964	Reject H0
		Hybrid	2.016.93	50.9123	39,602,730	−0.0511	Accept H0
	High	Genetic	2.007.63	55.54367	192,024,000		
		Pure	2.234.30	72.11433	7517,320	3.1432	Accept H0
		Hybrid	2.013.77	44.64951	158,410,920	0.1374	Accept H0
T-n100-k1	Low	Genetic	25.783.27	855.4517	18,753,750		
		Pure	33.493.27	2713.778	19,090,920	2.8411	Reject H0
		Hybrid	26.370.27	972.3565	15,283,500	0.6037	Accept H0
	Medium	Genetic	23.450.23	832.8231	75,007,500		
		Pure	32.350.37	1956.605	27,630,330	4.5488	Reject H0
		Hybrid	23.435.40	796.45	60,003,330	−0.0186	Accept H0
	High	Genetic	23.028.47	749.5474	300,030,000		
		Pure	**Results not obtained due to processing time**				
		Hybrid	22.419.73	964.6391	240,013,320	−0.6311	Accept H0
E-n101-k8	Low	Genetic	1.016.83	60.83479	18,753,750		
		Pure	2.720.70	79.82695	19,090,920	21.3445	Reject H0
		Hybrid	1.022.93	40.38724	15,283,500	0.1510	Accept H0
	Medium	Genetic	920.73	42.61215	76,060,530		
		Pure	1.116.50	55.26346	16,502,723	3.5424	Reject H0
		Hybrid	948.47	30.87365	61,133,610	0.8983	Accept H0
	High	Genetic	904.27	28.34776	306,060,300		
		Pure	**Results not obtained due to processing time**				
		Hybrid	917.17	35.47401	244,534,440	0.3636	Accept H0

Table 7
Comparison of results obtained from Talbis's algorithm and the proposed algorithm.

Problem	Algorithm	Minimum	Average	Standard Deviation	Maximum	Number Evaluations
Gr17 Opt.: 2085	Talbi	2085	2099.6	15.69	2120	208,000
	Pure	2085	2106.6	21.70	2154	194,764
	Hybrid	2085	2093.5	9.85	2116	207,816
Gr21 Opt.: 2707	Talbi	2707	2755.2	54.18	2851	288,000
	Pure	2709	2827.8	84.62	3044	271,396
	Hybrid	2707	2744.9	49.65	2833	260,270
Gr24 Opt.: 1272	Talbi	1272	1290.0	14.00	1314	640,000
	Pure	1272	1327.1	43.55	1411	475,364
	Hybrid	1272	1303.5	31.75	1366	540,310

Table 8
Parameters used in the "pure" version.

Problem	NQua	NObs	NGer	ε	TMut.
Gr17	2	8	1300	0.017	0.75
Gr21	2	8	1800	0.014	0.65
Gr24	8	2	4000	0.086	0.35

Table 9
Parameters used in the "hybrid" version.

Problem	NQua	NObs	NGer	ε	NCla	NGerc	TCroc	TMutc
Gr17	1	4	1000	0.03	17	1000	0.25	0.15
Gr21	1	5	1000	0.02	21	1000	0.6	0.15
Gr24	1	6	1000	0.01	24	2000	0.25	0.3

where p distinct products may be produced. The objective is to establish when each product must be produced within a planning horizon of n days. It is assumed that the line is operational for 24 h a day without interruption, with a nominal speed of unities per hour. Each product has an efficiency factor with respect to this nominal speed – production speed is the result of the efficiency factor multiplied by the nominal speed – and a predicted daily sale. There is an initial stock and a stocks policy, expressed by minimum and maximum quantities that must remain at the end of each day for every product.

For technical reasons, a product that enters the sequencing line must be produced continuously in multiples of h hours. This is determined by the production window. If, for example, $h = 2$ h, an item may be produced for 6 h, leave the production line and return later to be produced again for 4 h.

An aspect that makes this problem difficult to solving is the time needed for preparing the line after production of one item finishes and that of another item starts. This time depends on the nature of the two products.

The problem solution should contain the sequence of items to be produced and the time taken to produce each of them, so as to guarantee that sales and the stocks policy are attended to. Preparation times and production speed must obviously be respected. If some sale is lost or a stocks policy is not respected, solutions that do not satisfy the problem restrictions will be penalized. It is interesting to note that the best possible solutions are those which result in criterion functions with value zero, indicating that no so-

Table 10
Global data.

Number of products (p)	22
Planning horizon (n)	7 days
Production window (h)	4 hours
Line speed	25,000 units/hour
Cost of sale loss ($/fraction)	1000/fraction (if the loss is 100%, cost is 1000; if it is 50%, cost will be 500, etc)
Cost for not reaching minimum stock ($/fraction)	100/fraction
Cost for exceeding maximum stock ($/fraction)	1/fraction

Table 11
Product-dependent data.

Product	Sale day	Initial stock	Minimum stock	Maximum stock	Efficiency factor
1	19,447	77,784	77,788	350,046	0.5
2	10,608	42,432	42,432	190,944	0.5
3	672	2688	2688	20,160	0.5
4	6616	33,080	33,080	52,928	0.5
5	493	1972	1972	14,790	0.5
6	15,600	31,200	31,200	436,800	0.5
7	3560	7118	7120	99,680	0.5
8	8169	16,336	16,338	228,732	0.5
9	49,064	98,126	98,128	1,373,792	0.5
10	14,139	28,278	28,278	395,892	0.5
11	15,616	31,230	31,232	437,248	0.5
12	3499	10,494	10,497	101,471	0.5
13	4008	8016	8016	112,224	0.5
14	305	1830	1830	5795	0.5
15	3721	22,326	22,326	70,699	0.5
16	264	1578	1584	5016	0.5
17	35,639	178,195	178,195	285,112	0.5
18	9726	9726	9726	9726	0.5
19	1565	12,512	12,520	54,775	0.5
20	447	4917	4917	16,539	0.5
21	926	11,112	11,112	35,188	0.5
22	2346	11,725	11,730	70,380	0.5

Table 12
Line preparation time for each pair of products.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	1	1	6	1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
2	1	0	1	6	1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
3	1	1	0	6	1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
4	6	6	6	0	6	6	6	6	6	6	6	6	6	6	6	6	1	6	6	6	6	6
5	1	1	1	6	0	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
6	6	6	6	6	6	0	4	1	6	6	6	6	6	1	1	1	6	6	4	4	1	1
7	6	6	6	6	6	4	0	4	6	6	6	6	6	4	4	4	6	6	1	1	4	4
8	6	6	6	6	6	1	4	0	6	6	6	6	6	1	1	1	6	6	4	4	1	1
9	6	6	6	6	6	6	6	6	0	4	1	1	1	6	6	6	6	6	6	6	6	6
10	6	6	6	6	6	6	6	6	4	0	4	4	4	6	6	6	6	6	6	6	6	6
11	6	6	6	6	6	6	6	6	1	4	0	1	1	6	6	6	6	6	6	6	6	6
12	6	6	6	6	6	6	6	6	1	4	1	0	1	6	6	6	6	6	6	6	6	6
13	6	6	6	6	6	6	6	6	1	4	1	1	0	6	6	6	6	6	6	6	6	6
14	6	6	6	6	6	1	4	1	6	6	6	6	6	0	1	1	6	6	4	4	1	1
15	6	6	6	6	6	1	4	1	6	6	6	6	6	1	0	1	6	6	4	4	1	1
16	6	6	6	6	6	1	4	1	6	6	6	6	6	1	1	0	6	6	4	4	1	1
17	6	6	6	1	6	6	6	6	6	6	6	6	6	6	6	6	0	6	6	6	6	6
18	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	0	6	6	6	6
19	6	6	6	6	6	4	1	4	6	6	6	6	6	4	4	4	6	6	0	1	4	4
20	6	6	6	6	6	4	1	4	6	6	6	6	6	4	4	4	6	6	1	0	4	4
21	6	6	6	6	6	1	4	1	6	6	6	6	6	1	1	1	6	6	4	4	0	1
22	6	6	6	6	6	1	4	1	6	6	6	6	6	1	1	1	6	6	4	4	1	0

lution has been inhibited and, consequently, sales and stock conditions have been preserved.

Tables 10 and 11 below show the detailed configuration of the actual line scheduling problem. Table 10 presents the data that do not depend on the products – called Global data, whereas those which are product-dependent can be seen in Table 11.

Additionally, there are data that depend on a pair of products. They are expressed in hours in Table 12 and refer to the line preparation times, as explained before.

The initial stock is considered the same as the minimum stock. Therefore, a solution that satisfies all conditions regarding sales and stock will hardly be obtained, because it will be less likely to be able to produce all daily sale for all products in only one day.

Three levels of computational effort are again considered for the traditional order-based genetic algorithm, starting with 1,000,000 evaluations at the low level. This number is doubled and doubled again to characterize the medium and high levels, respectively. The proposed algorithms (“pure” and “ hybrid” versions) were exe-

Table 13
Comparison of the minimum values obtained by the proposed algorithm (pure and hybrid) and by the traditional order-based genetic algorithm for the line scheduling problem.

Comp. Effort	Algorithm	Min	Average	Max	Number Evaluations
Low	Genetic	6898.51	10345.1	13848.29	10,002,011
	Pure	7368.21	8427.61	9869.83	**9,514,028**
	Hybrid	**6522.05**	**8195.46**	**9853.05**	10,002,000
Medium	Genetic	6877.2	9168.84	10415.06	20,004,011
	Pure	7371.2	**8206.43**	**8872.8**	**19,010,490**
	Hybrid	**6514.19**	8784.28	11248.09	20,004,000
High	Genetic	6810.29	9055.59	10447.88	40,008,011
	Pure	7238.06	8133.95	8860.24	**16,748,696**
	Hybrid	**6305.9**	**7428.53**	**7984.6**	40,008,000

cuted so as to guarantee that the maximum number of evaluations, for each computational effort level, is the same as the traditional order-based genetic algorithm.

The traditional order-based genetic algorithm has been again implemented using roulette for individuals selection and 10% elitism. The mutation operator is the same as the permutation problem, but, in this case, the ordering crossover has been selected (Gen, & Cheng, 1997).

4.2.1. Comparative results

The same 110 combinations of crossover and mutation rates have been considered in this case study for the traditional genetic algorithm.

A parameter *Pot* is now included in the pure version of the algorithm, with the objective of reducing the weight of a bad solution when updating the quantum individual. Consider $FSol(t)$ and $FGer(t)$ as the values of the criterion function for the best solutions obtained until and at generation t, respectively. The $\varepsilon(t)$ for updating the quantum individual at generation t is:

$$\varepsilon(t) = \varepsilon\left(\frac{FSol(t)}{FGer(t)}\right)^{Pot} \tag{5-1}$$

Supposing that $FGer(t)$ is the value for a bad solution, then $FGer(t) > FSol(t)$, i.e. $\frac{FSol(t)}{FGer(t)} < 1$ ($\frac{FSol(t)}{FGer(t)}$ is always equal to or less than 1). Therefore, $(\frac{FSol(t)}{FGer(t)})^{Pot}$ falls as Pot increases.

For the "pure" version of the algorithm, the following combinations of parameters have been considered: $\varepsilon \in \{0{,}001;\ 0{,}003;\ 0{,}005;\ 0{,}007;\ 0{,}009\}$, $TMut \in \{0;\ 0{,}2;\ 0{,}4\}$ and $Pot \in \{4;\ 8\}$. In addition, the pair (*NGer, Nobs*) can assume the following values: {(250000,4); (500000,2)}. Thus, 60 ($5 \times 3 \times 2 \times 2$) different combinations have been evaluated. The computational effort was determined by the number of quantum individuals used (1 for the low level, 2 for the medium level and 4 for the high level).

For the "hybrid" version, the traditional order-based genetic algorithm is activated only once when the number of generations is reached. Again, since the objective is to generate, from the quantum individuals, an initial population for the genetic algorithm, the quantum individual should not be close to saturation when *Nger* is reached. Therefore, parameter ε was fixed at 0,001 and the saturation limit was taken as 0,99. Parameters *Pot* and *Tmut* were considered as zero in the hybrid version, since they are not relevant in this case. Either for the part that makes use of quantum individuals or for the traditional order-based genetic algorithm, $NGer = NGerc = 5000$. Crossover (*TCroc*) and mutation rates (*TMutc*) assumed values in the sets {0,1; 0,25; 0,4; 0,55; 0,7; 0,85; 1} and {0; 0,25; 0,5; 0,75; 1,0} respectively. Finally, the pairs *NObs* and *NCla* were: (6, 194), (12, 188) and (18, 182). A total of 105 combinations ($7 \times 5 \times 3$) were then analyzed.

It should be noted that both in the traditional order-based genetic algorithm and in the hybrid version the number of evaluations is defined a priori (unless the quantum individual saturates, but this must be avoided). On the other hand, in the pure version the number of evaluations will depend on the saturation of quantum individuals.

Table 13 presents, again, the minimum, average and maximum values of the best result obtained at the end of the 10 experiments of each different configuration of all algorithms evaluated.

As can be seen from Table 13, the hybrid version of the proposed quantum inspired order-based genetic algorithm provides better minimum solutions in all computational effort levels.

Although the minimum values obtained by the pure version are not as good as those for the other two algorithms, the opportunity for reducing the number of criterion function evaluations might be considered in applications where the cost for computing the criterion function is critical. For the high computational effort level, the pure version of QIEA-O performs well with around 42% less number of evaluations than those of the other two algorithms.

5. Conclusions

This paper presented a new algorithm for solving order optimization problems, based on evolutionary computation and quantum-inspired computing. The practical application of order optimization problems is very extensive and such problems have several solution techniques. The existence of several techniques indicates the lack of an algorithm that is clearly superior to the others and whose use always ensure the best result for any order optimization problem. In this context, the proposed algorithm is a viable alternative to solve such problems. This algorithm is applicable to any order problem regardless of the method for calculating the criterion function and the elements of universe *U*. Among the existing techniques based on evolutionary computation (Genetic Algorithms, Ant Colony, Particle Swarm, Simulated Annealing, Tabu Search, etc.), those based on genetic algorithms were chosen as the standard for comparing the results obtained with the proposed algorithm. Seven vehicle routing problems and three traveling salesman problems have been considered, with sizes ranging between 33 and 101 towns and between 1 and 10 vehicles. Furthermore, a problem based on a real case of production design was also considered. Two versions were considered for the proposed algorithm, called "pure" and "hybrid". In the "pure" version, only the solutions obtained through the observation of quantum individuals are contemplated, whereas in the "hybrid" version, the solutions

obtained from the quantum individuals form the initial population of a traditional order-based genetic algorithm.

The results obtained with the proposed algorithms were evaluated in terms of the average and best solution obtained and the total number of evaluations. Based on these indicators, the results for the quantum-inspired order-based genetic algorithm were compared with those provided by a traditional order-based genetic algorithm. In terms of quality of solution – based on the criterion function – the hybrid version of the proposed algorithm provided better results than those obtained with the traditional order-based genetic algorithm, especially for the line scheduling problem. The results of the pure version were worse with regard to criterion function. However, it is noteworthy that saturation of the quantum individual determines a stopping criterion, causing the evolutionary process to terminate before reaching the maximum number of generations specified. Therefore, the pure version of the algorithm provides a reasonably good solution in terms of the criterion function with a much smaller number of evaluations.

Acknowledgments

The authors would like to thank the Brazilian Agencies CNPq and FAPERJ for their financial support.

References

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2006). *The traveling salesman problem: A computational study*. New Jersey: Princeton University Press.

Babu, S. G. S., Das, D. B., & Patvardhan, C. (2008). Real-parameter quantum evolutionary algorithm for economic load dispatch. *IET Generation, Transmission and Distribution, 2*(January(1)) 2008.

Bayran, H., & Sahin, R. (2013). A new simulated annealing approach for travelling salesman problem. *Mathematical and Computational Applications, 18*(3), 313–322.

Bjarnadóttir, A. S. (2004). *Solving the vehicle routing problem with genetic algorithms* Master Thesis. Informatics and Mathematical Modelling. Technical University of Denmark.

Bin, Y., Zhong-Zhen, Y., & Baozhen, Y. (2009). An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research, 196*, 171–176.

Breedam, A. V. (1995). Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research, 86*, 480–490.

Chaturvedi, J. (2013). Application of quantum evolutionary algorithm to complex timetabling problem. *Open Science Repository Computer and Information Sciences*, e70081951 Online (open-access).

Chung, C. Y., Yu, H., & Wong, K. P. (2011). An advanced quantum-inspired evolutionary algorithm for unit commitment. *IEEE Transactions on Power Systems, 6*(May(2)) 2011.

Cruz, A. V. A., Vellasco, M. B. R., & Pacheco, A. C. (2008). Quantum-inspired evolutionary algorithm for numerical optimization. In C. Leandro dos Santos, & M. Luiza de Macedo (Eds.), *Quantum inspired intelligent systems*. In *Studies in computational intelligence: 121* (pp. 115–132). Springer Berlin Heidelberg.

Dias, D. M., & Pacheco, M. A. C. (2012). Quantum-inspired linear genetic programming as a knowledge management system. *The Computer Journal, 56*(9), 1043–1062.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *BioSystems, 43*, 73–81.

Falcone, J. L., Chen, X., & Hamad, G. G. (2013). The traveling salesman problem in surgery: Economy of motion for the FLS peg transfer task. *Surgical Endoscopy, 27*, 1636–1641.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P., Reis, M., Uchoa, E., et al. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming, 106*(3), 491–511.

Gen, M., & Cheng, R. (1997). Genetic algorithms and engineering design. *Wiley series in engineering design and automation*. New York: John Wiley and Sons.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science, 40*(10), 1276–1290.

Gopalakrishnan, B., & Johnson, E. L. (2005). Airline crew scheduling state-of-the-art. *Annals of Operations Research, 140*, 305–337.

Han, K., & Kim, J. H. (2000). Genetic quantum algorithm and its application to combinatorial optimization problem. In *Proceedings of the 2000 congress on evolutionary computation* (pp. 1354–1360).

Han, K.-H., & Kim, J.-H. (2002). Quantum-inspired evolutionary algorithm for a class of combina-torial optimization. *IEEE Transactions on Evolutionary Computation 2002, 6*, 580–593.

Herrero, R., Ramos, J.J., & Guimarans, D. (2010). Lagrangean metaheuristic for the travelling salesman problem. OR52 Extended Abstracts, pp. 204–211.

Jang, J. S., Han, K. H., Kim, J. H., Yu, H., & Wong, K. P. (2004). Face detection using quantum-inspired evolutionary algorithm. In *IEEE evolutionary computation, 2004. CEC2004: 2* (pp. 2100–2106).

Kim, Y.-H., & Kim, J.-H. (2009). Multiobjective quantum-inspired evolutionary algorithm for fuzzy path planning of mobile robot. In *2009 IEEE congress on evolutionary computation (CEC 2009)* (pp. 1185–1192).

Karapetyan, D., & Gutin, G. (2011). Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem. *European Journal of Operational Research, 208*(3), 221–232.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Neural Networks, 62*, 1942–1948.

Li, Z., Xu, B., Yang, L., Chen, J., & Li, K. (2009). Quantum evolutionary algorithm for multi-robot coalition formation. In *GEC 2009, proceedings of the first ACM/SIGEVO summit on genetic and evolutionary computation* (pp. 295–302).

Luo, Z., Wang, P., Li, Y., Zhang, W., Tang, W., & Xiang, M. (2008). Quantum-inspired evolutionary tuning of SVM parameters. *Progress in Natural Science, 18*(4), 475–480 April 2008.

Malek, M., Guruswamy, M., Pandya, M., & Owens, H. (1989). Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research, 21*, 59–84.

Marinakis, Y., & Marinaki, M. (2010). A hybrid genetic – particle swarm optimization algorithm for the vehicle routing problem. *Expert Systems with Applications, 37*, 1446–1455.

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation, 8*(1), 3–30.

Moore, M., & Narayanan, A. (1995). *Quantum-inspired computing*. University of Exeter Computer Science Old Library.

Narayanan, A., & Moore, M. (1996). Quantum-inspired genetic algorithms. In *Evolutionary computation – proceedings of IEEE international conference on* (pp. 61–66).

Nazif, H., & Lee, L. S. (2012). Optimised crossover genetic algorithm for capacitated vehicle routing problem. *Applied Mathematical Modelling, 36*, 2110–2117.

Ning, T., Guo, C., & Chen, R. (2015). A novel method for dynamic vehicle routing problem. *The Open Cybernetics & Systemics Journal, 9*, 2254–2258 *2015*.

Özbakır, L., & Tapkan, P. (2011). Bee colony intelligence in zone constrained two-sided assembly line balancing problem. *Expert Systems with Applications, 38*(11), 11947–11957.

Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review, 33*(1), 60–100.

Pichpibul, T., & Kawtummachai, R. (2013). A heuristic approach based on clarke-wright algorithm for open vehicle routing problem. *The Scientific World Journal, 2013*, 11 Article ID 874349.

Pinho, A. G., Vellasco, M. M. B. R., & Cruz, A. V. A. (2009). A new model for credit approval problems: A quantum-inspired neuro-evolutionary algorithm with binary-real representation. In *Proceedings of the 2009 world congress on nature & biologically inspired computing (NaBIC)* (pp. 445–450).

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*, 1985–2002.

Rani, K., & Kumar, V. (2014). Solving travelling salesman problem using genetic algorithm based on heuristic crossover and mutation operator. *International Journal of Research in Engineering & Technology, 2*(2), 27–34.

Roy, S., Panja, U. K., & Sardar, N. K. (2014). Efficient technique to solve travelling salesman problem using genetic algorithm. *International Journal of Computer Engineering & Technology, 5*(3), 132–137.

Rieffel, E., & Polak, W. (2000). An introduction to quantum computing for non–physicists. *Journal ACM Computing Surveys, 32*(3), 300–335.

Setia, R., & Hans, R. K. (2012). Quantum inspired evolutionary algorithm for optimization of hot extrusion process. *International Journal of Soft Computing and Engineering (IJSCE), 2*(November(5)) 2012.

Sbihi, A., & Eglese, R. W. (2007). Combinatorial optimization and green logistics. *4OR, 5*, 99–116.

Silveira, L. R., Tanscheit, R., & Vellasco, M. M. B. R. (2012). Quantum-inspired genetic algorithms applied to ordering combinatorial optimization problems. *IEEE Congress on Evolutionary Computation (CEC)*, 1341–1347.

Silveira, L. R. (2014). *Algoritmo genético de ordem com inspiração quântica* PhD Thesis. Pontificia Universidade Católica do Rio de Janeiro.

Talbi, H., Draa, A., & Batouche, M. (2004). A new quantum-inspired genetic algorithm for solving the travelling salesman problem. *IEEE International Conference on Industrial Technology*, 1192–1197.

Vellasco, M. B. R., Cruz, A., & Pinho, A. (2010). Quantum-inspired evolutionary algorithms applied to neural network modeling. In Joan Aranda, & Sebastià Xambó (Eds.), *IEEE world congress on computational intelligence (WCCI), plenary and invited lectures* (pp. 125–150).

Vlachogiannis, J. G., & Lee, K. Y. (2008). Quantum-inspired evolutionary algorithm for real and reactive power dispatch. *IEEE Transactions on Power Systems, 23*(November(4)) 2008.

Wang, L., Wu, H., & Zheng, D.-Z. (2005). A quantum-inspired genetic algorithm for scheduling problems. In *Advances in natural computation, of the Series lecture notes in computer science: Vol. 3612* (pp. 417–423). Springer Berlin Heidelberg.

Wu, Y., Ji, P., & Wang, T. (2008). An empirical study of pure genetic algorithm to solve the capacitated vehicle routing problem. In *ICIC international 2008* (pp. 41–45).

Xiao, J., Yan, Y., Lin, Y., Yuan, L., & Zhang, J. (2008). A quantum-inspired genetic algorithm for data clustering. In *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)* (pp. 1513–1519).

Zhang, G. (2010). Quantum-inspired evolutionary algorithms: A survey and empirical study. *Journal of Heuristics, 17*(3), 303–351.