European Journal of Operational Research 231 (2013) 22–33

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

An efficient evolutionary algorithm for the ring star problem ☆

Herminia I. Calvete [a,*], Carmen Galé [b], José A. Iranzo [a]

[a] Dpto. de Métodos Estadísticos, IUMA, Universidad de Zaragoza, Pedro Cerbuna 12, 50009 Zaragoza, Spain
[b] Dpto. de Métodos Estadísticos, IUMA, Universidad de Zaragoza, María de Luna 3, 50018 Zaragoza, Spain

ARTICLE INFO

Article history:
Received 7 November 2012
Accepted 10 May 2013
Available online 23 May 2013

Keywords:
Ring star problem
Median cycle problem
Evolutionary algorithm
Bilevel programming

ABSTRACT

This paper addresses the ring star problem (RSP). The goal is to locate a cycle through a subset of nodes of a network aiming to minimize the sum of the cost of installing facilities on the nodes on the cycle, the cost of connecting them and the cost of assigning the nodes not on the cycle to their closest node on the cycle. A fast and efficient evolutionary algorithm is developed which is based on a new formulation of the RSP as a bilevel programming problem with one leader and two independent followers. The leader decides which nodes to include in the ring, one follower decides about the connections of the cycle and the other follower decides about the assignment of the nodes not on the cycle. The bilevel approach leads to a new form of chromosome encoding in which genes are associated to values of the upper level variables. The quality of each chromosome is evaluated by its fitness, by means of the objective function of the RSP. Hence, in order to compute the value of the lower level variables, two optimization problems are solved for each chromosome. The computational results show the efficiency of the algorithm in terms of the quality of the solutions yielded and the computing time. A study to select the best configuration of the algorithm is presented. The algorithm is tested on a set of benchmark problems providing very accurate solutions within short computing times. Moreover, for one of the problems a new best solution is found.

1. Introduction

The Ring Star Problem (RSP) arises in telecommunications network design problems where a subset of nodes is selected for installing concentrators. The selected nodes are interconnected by a ring network (this is known as ring topology), whereas the remaining nodes are linked to their closest concentrator (this is known as star topology). The RSP can also model location–allocation problems occurring in logistics where some retailer locations are used as small depots served from a central depot in a single-vehicle route. The remaining retailers are directly served from their nearest small depot.

Let $G = (V, E \cup A)$ be a mixed graph, where $V = \{0, 1, \ldots, n\}$ is the node set, $E = \{[i,j]: i, j \in V\}$ is the edge set and $A = \{(i,j): i, j \in V\}$ is the arc set. Node 0 is a distinguished node which is referred to as the depot or the root. Edges in E refer to undirected links which are used to form the ring structure and arcs in A refer to directed links used in the star structure. We assume that there is a nonnegative facility cost p_i associated with each node i. This may represent, for instance, the cost of placing a facility at node i. There is also a nonnegative ring cost c_{ij} associated with each edge $[i,j]$, representing the cost of connecting nodes i and j, and a nonnegative assignment cost d_{ij} associated with each arc (i,j), referring to the cost of node i being connected to node j. The RSP consists of selecting a subset of nodes $V' \subseteq V$, including the depot, where facilities are installed and interconnected by a cycle structure. The remaining nodes are each connected to one of the facilities (see Fig. 1). The goal is to minimize the total cost, being the sum of the cost of placing the facilities at the nodes of the cycle plus the cost of the ring connections and the assignment costs. The cycle connection cost is the sum of all edge costs on the cycle. The assignment cost is defined as $\sum_{i \in V \setminus V'} \min_{j \in V'} d_{ij}$.

To formulate the RSP, we define

$$z_i = \begin{cases} 1, & \text{if } i \in V \text{ is on the cycle} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if edge } [i,j] \in E \text{ is on the cycle} \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1, & \text{if node } i \in V \text{ is assigned to node } j \text{ on the cycle,} \\ & \qquad (i,j) \in A \\ 0, & \text{otherwise} \end{cases}$$

In order to simplify the notation, we denote $\{z_i, i \in V; x_{ij}, [i,j] \in E; y_{ij}, (i,j) \in A\}$ by $\{z,x,y\}$.

Then, the RSP can be formulated as the following binary problem:

☆ This research work has been funded by the Gobierno de Aragón under grant E58 (FSE) and by Ibercaja under grant UZ2011-CIE-01.
* Corresponding author. Tel.: +34 976 762210.
E-mail addresses: herminia@unizar.es (H.I. Calvete), cgale@unizar.es (C. Galé), joseani@unizar.es (J.A. Iranzo).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.05.013

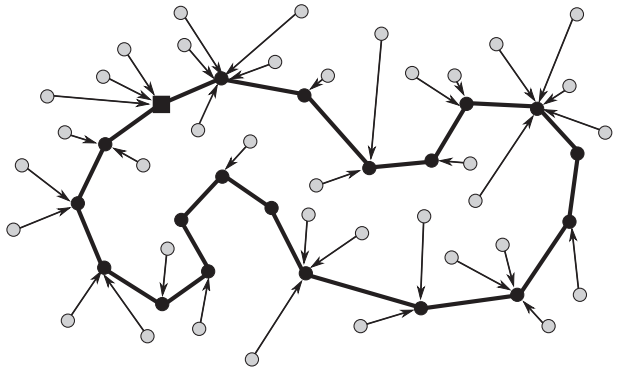


Fig. 1. A feasible solution of the RSP.

$$\min_{z,x,y} \sum_{i\in V} p_i z_i + \sum_{[i,j]\in E} c_{ij} x_{ij} + \sum_{(i,j)\in A} d_{ij} y_{ij} \tag{1a}$$

$$\text{subject to} \quad \sum_{[i,j]\in E} x_{ij} = 2z_i, \quad \forall i \in V \tag{1b}$$

$$\sum_{(i,j)\in A} y_{ij} = 1 - z_i, \quad \forall i \in V \tag{1c}$$

$$\sum_{[i,j]\in E(S)} x_{ij} \leqslant |S| - 1, \quad \forall S \subseteq V \setminus \{0\}, \quad S \neq \emptyset \tag{1d}$$

$$z_0 = 1 \tag{1e}$$

$$z_i \in \{0,1\}, \quad \forall i \in V \tag{1f}$$

$$x_{ij} \in \{0,1\}, \quad \forall [i,j] \in E \tag{1g}$$

$$y_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{1h}$$

where $E(S) = \{[i,j] \in E : i, j \in S \subseteq V\}$. In this formulation, the objective function (1a) minimizes the total cycle and assignments costs, and the cost of placing the facilities at the nodes. Constraints (1b) enforce that the degree of each node i is 2 if and only if it belongs to the cycle. Constraints (1c) ensure that either each node i is on the cycle or is assigned to a single node j on the cycle. Constraints (1d) are the well-known subtour elimination constraints limiting the number of cycles to one. Constraint (1e) ensures that the depot is on the cycle. Constraints (1f)–(1h) guarantee that all variables are binary.

The RSP is also known as the first version of the Median Cycle Problem (MCP1). Under this name, Moreno Pérez et al. [22] use the model to test a heuristic that combines variable neighborhood search and tabu search using a set of benchmark problems based on Traveling Salesman Problem (TSP) instances from TSPLIB [24] involving between 50 and 200 nodes. In order to provide an optimal solution of the RSP, the algorithms developed in the literature mainly consider branch and cut techniques. Labbé et al. [17] formulate the problem as a mixed integer linear program, which uses connectivity constraints instead of constraints (1d), and relaxes some integer requirements of variables $\{z_i\}_{i\in V}$ and $\{y_{ij}\}_{(i,j)\in A}$ in formulation (1). After a polyhedral analysis of the problem, they propose a branch-and-cut algorithm and apply it to solve the benchmark problems and some randomly generated instances with a number of nodes between 10 and 300. Kedad-Sidhoum and Nguyen [16] propose a new formulation of the RSP based on chains and use it to develop a new branch-and-cut approach. A branch-and-cut algorithm is also proposed by Simonetti et al. [26], who reformulate the RSP as a minimum Steiner arborescence problem. They provide an optimal solution for some benchmark instances not solved in [17]. In summary, an optimal solution has been achieved for 120 out of the 124 benchmark instances considered in the literature [17,26].

Besides the above mentioned heuristic, other heuristic algorithms have been proposed in the literature. Renaud et al. [25] propose a multistart greedy add heuristic and a random keys evolutionary algorithm. Both algorithms are tested using only the benchmark instances which were exactly solved in [17] and proved to be quite efficient compared with [22]. Dias et al. [12] propose a heuristic that uses a general variable neighborhood search to improve the quality of the solution obtained by a greedy randomized adaptive search procedure. They only solve the benchmark instances having between 50 and 100 nodes, obtaining better results than [22] in some cases.

Other variants of the RSP have been introduced in the literature. Baldacci et al. [3] introduce the capacitated m-ring-star problem which consists of designing a set of m rings with bounded capacity that passes through the depot and through some transition points and/or customers, and then assigning each non-visited customer to a visited point or customer. A branch-and-cut approach is proposed to solve the problem. For the same problem, Naji-Azimi et al. [23] develop a heuristic algorithm which follows the scheme of the Variable Neighborhood Search and incorporates an Integer Linear Programming based improvement. Baldacci and Dell'Amico [2] generalize the above problem by allowing the existence of multiple depots. Finally, Liefooghe et al. [19] propose to consider individually on the one hand the costs of the ring and on the other hand the assignment costs. Based on this bi-objective formulation they propose different metaheuristics for solving the problem.

The goal of this paper is to propose an evolutionary algorithm for the RSP based on a new encoding scheme to handle which nodes are on the ring. This algorithm has been suggested by a new formulation of the RSP as a bilevel problem with three decision makers, a leader and two independent followers. The paper is organized as follows. In Section 2 the formulation of the RSP as a bilevel problem is proposed and the main results of the equivalence between both problems are proved. The algorithm is developed in Section 3. In Section 4 the computational performance of the procedure is evaluated using the benchmark instances dealt with in the literature. Finally, Section 5 concludes the paper with some final remarks.

2. A bilevel formulation for the RSP

Bilevel programming has been proposed for modeling hierarchical processes characterized by the existence of two decision levels. The decision makers at both levels of the hierarchy seek to optimize their individual objective functions and control their own set of decision variables. Due to the hierarchical structure of the process, the decision maker at the upper level of the hierarchy, also called the leader, aims to optimize his own objective function but anticipating within the optimization scheme the reaction of the decision maker at the lower level, also called the follower. In mathematical terms, the bilevel programming problem involves two optimization problems where the constraint region of the upper level optimization problem is implicitly determined by the lower level optimization problem. Bilevel programming is discussed in Bard [4], Colson et al. [10] and Dempe [11]. Some extensions of bilevel programming consider the existence of several decision makers at the lower decision level [6] or multiple objectives at each decision level [7,8].

In the RSP, the decision maker has to locate a simple cycle and assign the remaining nodes to the nodes on the cycle in an optimal way. In the bilevel programming formulation that we propose, the decision maker will share the decision process. This decision maker will act as a leader and delegate some of the decisions to two followers. The idea is that the leader will decide on the nodes of the cycle, but anticipating the reactions of both followers. Each follower, after receiving the leader's selection, solves his own problem. One follower will find the cycle by solving a traveling

salesman problem and the other follower will solve the problem of assigning the nodes not in the cycle to their closest node on the cycle. These followers are independent in the sense that there is no communication between them and they do not share any information. Taking into account the hierarchical structure of the bilevel problem, the leader maintains overall control of the process. Moreover, since the leader is assumed to anticipate the reactions of the followers, he will be able to choose his optimal strategy. Fig. 2 displays a scheme of this decision process.

In order to reformulate the RSP as a bilevel problem, we assume that variables $\{z_i\}_{i\in V}$ are controlled by the leader, variables $\{x_{ij}\}_{[i,j]\in E}$ are controlled by follower 1 and variables $\{y_{ij}\}_{(i,j)\in A}$ are controlled by follower 2. The bilevel reformulation of the RSP (BRSP) is:

$$\min_{z,x,y} \sum_{i\in V} p_i z_i + \sum_{[i,j]\in E} c_{ij}x_{ij} + \sum_{(i,j)\in A} d_{ij}y_{ij} \tag{2a}$$

$$\text{subject to} \quad z_0 = 1 \tag{2b}$$

$$z_i \in \{0,1\}, \quad \forall i \in V \tag{2c}$$

where, for every $\{z_i\}_{i\in V}$ fixed, $\{x_{ij}\}_{[i,j]\in E}$ solves the problem:

$$L_1(z): \quad \min_{x} \sum_{[i,j]\in E} c_{ij}x_{ij} \tag{2d}$$

$$\text{subject to} \sum_{[i,j]\in E} x_{ij} = 2z_i, \quad \forall i \in V \tag{2e}$$

$$\sum_{[i,j]\in E(S)} x_{ij} \leqslant |S| - 1, \quad \forall S \subseteq \{i \in V \setminus \{0\} : z_i = 1\}, \tag{2f}$$

$$x_{ij} \in \{0,1\}, \quad \forall [i,j] \in E \tag{2g}$$

and $\{y_{ij}\}_{(i,j)\in A}$ solves the problem:

$$L_2(z): \quad \min_{y} \sum_{(i,j)\in A} d_{ij}y_{ij} \tag{2h}$$

$$\text{subject to} \sum_{(i,j)\in A} y_{ij} = 1 - z_i, \quad \forall i \in V \tag{2i}$$

$$y_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{2j}$$

This is a binary bilevel problem with two independent followers, meaning that the objective function and the set of constraints of each follower only include the leader's variables and the individual follower's own variables. Bilevel problems with multiple independent followers have been addressed by Calvete and Galé [6] when all the functions involved are linear and there are no integrality constraints.

We now prove the main results concerning the equivalence between the RSP and the BRSP.

Lemma 1. *Let $\{\bar{z}_i\}_{i\in V}$ verify constraints (2b) and (2c). Let $\{\bar{x}_{ij}\}_{[i,j]\in E}$ and $\{\bar{y}_{ij}\}_{(i,j)\in A}$ be feasible solutions of problems $L_1(\bar{z})$ and $L_2(\bar{z})$, respectively. Then, $\{\bar{z},\bar{x},\bar{y}\}$ is a feasible solution of the RSP.*

Proof. It is obvious that constraints (1b), (1c), (1e)–(1h) are verified.

On the other hand, given $S \subseteq V\backslash\{0\}$, we define $S_0 = \{i \in S : \bar{z}_i = 0\}$ and $S_1 = \{i \in S : \bar{z}_i = 1\}$. Bearing in mind constraints (2e), $\bar{x}_{ij} = 0$ for all $[i,j] \in E$ and $i \in S_0$ or $j \in S_0$. Then, by applying (2f)

$$\sum_{[i,j]\in E(S)} \bar{x}_{ij} = \sum_{[i,j]\in E(S_1)} \bar{x}_{ij} \leqslant |S_1| - 1 \leqslant |S| - 1$$

This completes the proof. □

From the above lemma we can also conclude that every feasible solution of the BRSP provides a feasible solution of the RSP.

Lemma 2. *Let $\{\bar{z},\bar{x},\bar{y}\}$ be an optimal solution of the RSP. Then, it is a feasible solution of the BRSP.*

Proof. Since $\{\bar{z},\bar{x},\bar{y}\}$ is a feasible solution of the RSP, it verifies all constraints of problem (2), i.e. (2b), (2c), (2e)–(2g), (2i) and (2j). In order to show that it is a feasible solution of the BRSP, we need to prove that $\{\bar{x}_{ij}\}_{[i,j]\in E}$ is an optimal solution of problem $L_1(\bar{z})$ and $\{\bar{y}_{ij}\}_{(i,j)\in A}$ is an optimal solution of problem $L_2(\bar{z})$.

Let us first consider the problem $L_1(\bar{z})$. Assume that $\{\bar{x}_{ij}\}_{[i,j]\in E}$ is not an optimal solution of this problem. Then, there exists $\left\{x^*_{ij}\right\}_{[i,j]\in E}$, a feasible solution of $L_1(\bar{z})$ so that

$$\sum_{[i,j]\in E} c_{ij}x^*_{ij} < \sum_{[i,j]\in E} c_{ij}\bar{x}_{ij} \tag{3}$$

By applying Lemma 1, $\{\bar{z},x^*,\bar{y}\}$ is a feasible solution of the RSP and

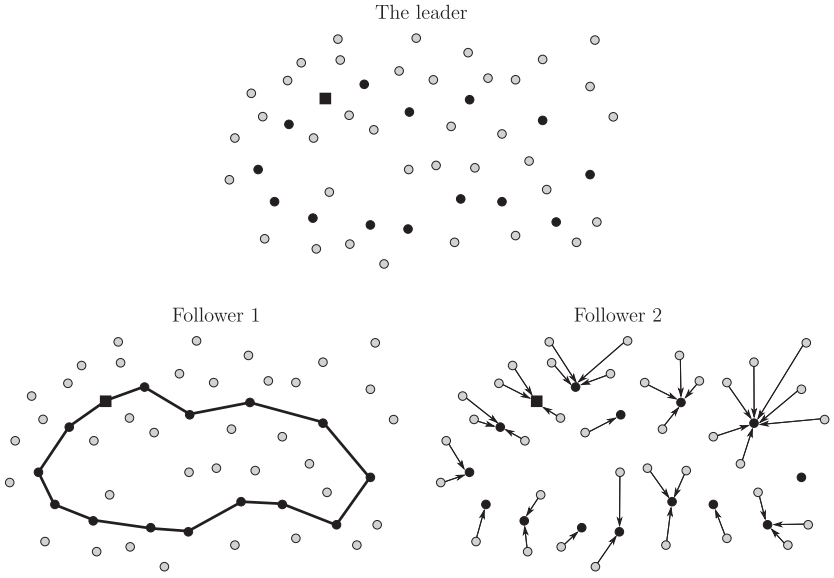


Fig. 2. Scheme of the bilevel approach of the RSP: The leader decides which nodes are to be included in the cycle. Follower 1 solves a TSP to find the cycle. Follower 2 optimally assigns the nodes not on the cycle.

$$\sum_{i\in V} p_i\bar{z}_i + \sum_{[i,j]\in E} c_{ij}x^*_{ij} + \sum_{(i,j)\in A} d_{ij}\bar{y}_{ij} < \sum_{i\in V} p_i\bar{z}_i + \sum_{[i,j]\in E} c_{ij}\bar{x}_{ij} + \sum_{(i,j)\in A} d_{ij}\bar{y}_{ij}$$

which contradicts that $\{\bar{z},\bar{x},\bar{y}\}$ is optimal for the RSP. The second part of the proof is analogous. □

Theorem 3. *The RSP and the BRSP provide the same optimal solution.*

Proof. Let $\{z^*,x^*,y^*\}$ be an optimal solution of the RSP. By applying Lemma 2, it is a feasible solution of the BRSP. Assume that it is not an optimal solution of this problem. Then, there exists a feasible solution of the BRSP $\{\bar{z},\bar{x},\bar{y}\}$ so that

$$\sum_{i\in V} p_i\bar{z}_i + \sum_{[i,j]\in E} c_{ij}\bar{x}_{ij} + \sum_{(i,j)\in A} d_{ij}\bar{y}_{ij} < \sum_{i\in V} p_iz^*_i + \sum_{[i,j]\in E} c_{ij}x^*_{ij} + \sum_{(i,j)\in A} d_{ij}y^*_{ij} \qquad (4)$$

Due to Lemma 1, $\{\bar{z},\bar{x},\bar{y}\}$ is a feasible solution of the RSP. Hence, (4) is in contradiction to the optimality of $\{z^*,x^*,y^*\}$.

Conversely, let $\{z^*,x^*,y^*\}$ be an optimal solution of the BRSP. By applying Lemma 1, it is a feasible solution of the RSP. Assume that it is not an optimal solution of the RSP. Then

$$\sum_{i\in V} p_i\bar{z}_i + \sum_{[i,j]\in E} c_{ij}\bar{x}_{ij} + \sum_{(i,j)\in A} d_{ij}\bar{y}_{ij} < \sum_{i\in V} p_iz^*_i + \sum_{[i,j]\in E} c_{ij}x^*_{ij} + \sum_{(i,j)\in A} d_{ij}y^*_{ij} \qquad (5)$$

where $\{\bar{z},\bar{x},\bar{y}\}$ is an optimal solution of the RSP. From Lemma 2, it follows that $\{\bar{z},\bar{x},\bar{y}\}$ is a feasible solution of problem BRSP. Hence, (5) is in contradiction to the optimality of $\{z^*,x^*,y^*\}$. □

In the following section we use these ideas about sharing the decision making process to develop an evolutionary algorithm which solves the BRSP and thus the RSP.

3. BBEA: A bilevel programming based evolutionary algorithm to solve the RSP

Evolutionary Algorithms (EAs) are adaptive heuristic search algorithms based on the principles of biological evolution. They use the ideas of reproduction and selection in populations to produce good solutions to complex optimization problems. Candidate solutions to the optimization problem play the role of individuals in biology. EAs were introduced and developed by Holland [14] and have been increasingly applied to solve a variety of problems [1,9,21]. EAs encode each potential solution as a string of symbols called a chromosome. Each position of a symbol in the chromosome is called a gene and its value is called the allele value. Then, crossover and mutation operators are applied to these structures so as to preserve critical information based on their fitness. The fitness gives each chromosome a score based on how well it performs, usually through the evaluation of the objective function of the optimization problem. An implementation of an EA starts with a randomly generated population of chromosomes whose fitness is evaluated. The population evolves to create offspring by combining chromosomes from the current population using a crossover operation and modifying them by using a mutation operation. Having evaluated the fitness of the new chromosomes, a selection operation allows some of the parents and offspring to survive to the next population. This process continues through successive iterations of the algorithm until a number of populations have been produced or a suitable solution has been found depending on the stopping criterion.

As far as we know, Renaud et al. [25] are the only researchers to have developed an EA to solve the RSP. In this algorithm, called the RKEA, each chromosome encodes a cycle, i.e. a feasible solution of the RSP, using the random keys encoding mechanism developed by Bean [5]. This encoding easily allows for the recombination of cycles. Moreover, no mutation operator is applied. Following the ideas presented in the above section, in the algorithm developed in this paper the chromosomes encode the upper level variables of the BRSP. From these values, a feasible solution of the BRSP can be obtained by solving both the followers' problems. Next, the algorithm proceeds by performing crossover, mutation, fitness evaluation and selection until the stopping condition is met.

In order to improve the performance of the algorithm, we have implemented the ideas on distributed genetic algorithms proposed by Tanese [27]. Hence, the population is divided into smaller subpopulations which evolve in an isolated manner producing offspring for the next generation. After a number of iterations, there is an information exchange operation as a result of which some selected individuals migrate to a different subpopulation. Moreover, while EA literature shows that on average the population improves over the iterations, sometimes the algorithm fails to generate near-optimal solutions and requires some form of intensification. Hence, we propose to use local search techniques to improve some randomly selected solutions. A sketch of the algorithm is shown in Fig. 3. Fig. 3a shows the steps of the algorithm and Fig. 3b displays the migration process.

3.1. Chromosome encoding and population handling

We encode the chromosome as a binary $|V|$-dimensional vector $C \in \{0,1\}^{|V|}$, so that for each $i \in V$

$$C_i = \begin{cases} 1, & \text{if node } i \text{ is on the cycle} \\ 0, & \text{otherwise} \end{cases}$$

Node 0 is always on the cycle, hence $C_0 = 1$.

Each chromosome $\overline{C}$ can be associated with a feasible solution $\{\bar{z},\bar{x},\bar{y}\}$ of the BRSP, and thus a feasible solution of the RSP. The upper level variables can be directly obtained from the chromosome: $\bar{z}_i = \overline{C}_i,\ i \in V$. Having fixed the upper level variables, we can compute the remaining variables by solving both lower level problems. Follower 1's problem is a TSP, thus it is difficult to solve. Since this procedure has to be applied to each chromosome generated by the algorithm, a compromise is needed between solution quality and computing time. Therefore, we have decided to use a procedure not very costly in terms of time and reasonable in terms of quality of the solution. We have selected a greedy algorithm based on least cost edges, followed by 2-opt local search. Moreover, with a probability of 0.5, 3-opt local search is also applied. Needless to say, other algorithms proposed in the literature for solving the TSP could also be applied [15,18]. Follower 2's problem is a very easy optimization problem. For each node not on the cycle, it selects the closest one on the cycle. Hence, it is solved to optimality.

The initial population is formed by randomly generated *IP* chromosomes. This population is divided into *P* subpopulations of size $PS = IP/P$. Let *MI* be the migration interval, i.e. the number of iterations between each migration. Then, each subpopulation evolves independently for *MI* iterations producing offspring via crossover, mutation and local search. Every *MI* iterations, the best individual of each subpopulation migrates to another subpopulation following the diagram shown in Fig. 3b.

3.2. Crossover, mutation and local search

The crossover operator represents the process by which chromosomes selected from a population are combined to form offspring which are potential members of a successor population. In the BBEA we apply a uniform crossover operator which enables the parent chromosomes to contribute the gene level. For each subpopulation of size *PS*, the crossover operator randomly selects

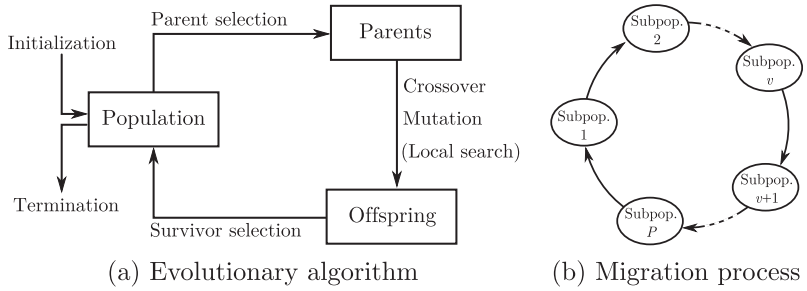
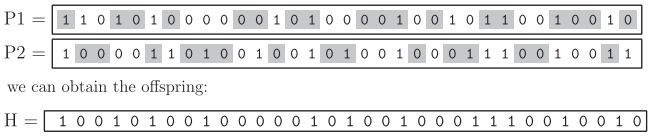


Fig. 3. Sketch of the BBEA.

PS pairs of parents and generates one offspring from each pair. Each gene of the offspring is selected from one of the parents with a probability of 0.5. For instance, from the parents:



Once offspring have been created, the mutation operator is applied to them. Each offspring is selected for the mutation operation with a probability of 0.5. After a chromosome has been selected, a gene is randomly selected and its allele value is switched.

In order to improve the quality of the populations which are generated in successive iterations, we propose to apply local search techniques to some of the feasible solutions associated with offspring, aiming to improve their quality measured by means of the objective function (1a). Each offspring is selected for the local search operation with a probability of 0.5. Let $\overline{C}$ be a selected offspring and $\{\bar{z}, \bar{x}, \bar{y}\}$ be the associated feasible solution of the BRSP. In the implementation of the algorithm, cycle reduction, cycle augmentation and node exchange [25] are successively applied insofar as they improve the objective function value of the solution obtained. Fig. 4 shows these techniques. Let $\{z^*, x^*, y^*\}$ be the feasible solution of the RSP obtained after applying the local search procedures. The offspring resulting from the local search operation is C^* so that $C_i^* = z_i^*$, $i \in V$ and its associated feasible solution is $\{z^*, x^*, y^*\}$.

3.3. Fitness evaluation and survivor selection

The quality of a chromosome is evaluated by its fitness. Hence, we define the fitness as the value of the objective function (1a) of the associated feasible solution of the RSP:

$$F(C) = \sum_{i \in V} p_i z_i + \sum_{[i,j] \in E} c_{ij} x_{ij} + \sum_{(i,j) \in A} d_{ij} y_{ij}$$

Finally, some of the parents and offspring are allowed to survive by applying the selection operator. Based on the chromosome fitness, we use the elitist strategy which keeps the best *PS* chromosomes in each subpopulation from one iteration to the next (without repetition).

After the stopping condition of the algorithm is met, the chromosome with the least fitness value will be selected and the associated feasible solution of the RSP will be provided as the solution of the RSP.

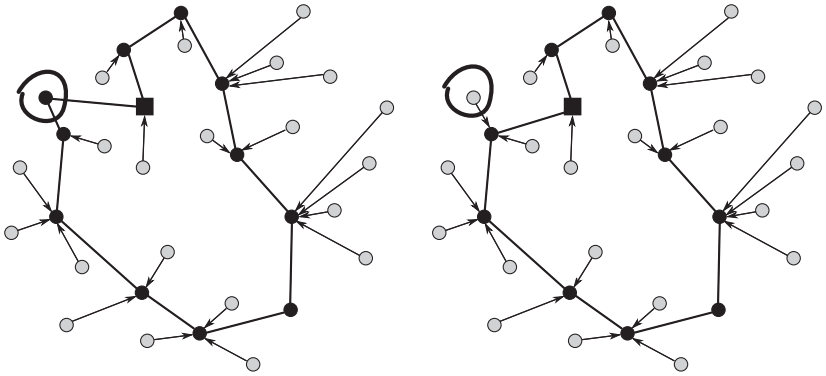
Having found the best solution provided by the algorithm, we have considered the possibility of applying an additional improvement in the implementation of the algorithm. This final step consists of applying version 2.0.6 of the implementation by Helsgaun [13] of the Lin and Kernighan heuristic [20] to the nodes in the cycle in order to obtain a possible better arrangement of the cycle. This implementation can be downloaded from http://www.akira.ruc.dk/~keld/research/LKH/. As we will explain in the next section, this final improvement appears to be interesting in the case that the optimal solution of the RSP consists of a cycle with almost all the nodes in the network, but not otherwise.

4. Computational results

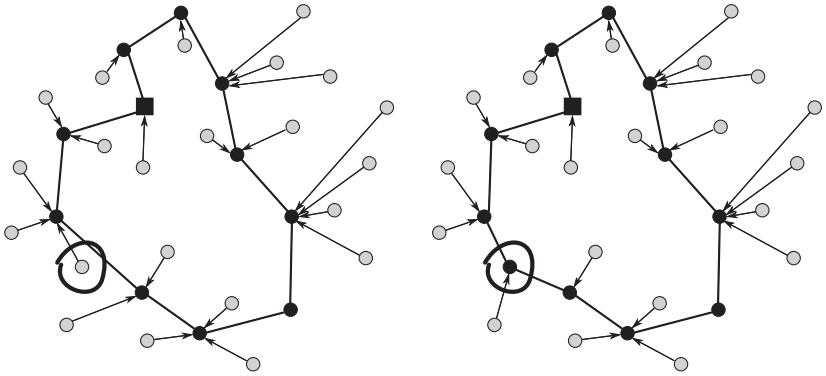
In order to analyze the performance of the BBEA, we have conducted a computational experiment divided into two parts. In the first part of the experiment, we have studied the effect of different parameters of the algorithm on the quality of the solution provided by the BBEA. Having selected the best value of the parameters, in the second part of the study we have compared the solutions provided by the BBEA with the optimal solutions available in the literature. Also in the second part, we have compared the BBEA with the RKEA, the evolutionary algorithm proposed by Renaud et al. [25]. The numerical experiments have been performed on a PC Intel® Core™ I7-3820 CPU at 3.6 gigahertz × 8 having 32 gigabytes of RAM under Ubuntu Linux 12.04 LTS. The code has been written in C++, GCC 4.6.3.

The performance of the algorithm has been tested on the above mentioned RSP benchmark problems. There are 124 problems which involve between 50 and 200 nodes. They are derived from the EUC-2D and EXPLICIT TSPLIB problems eil51 to kroB200 by modifying the original distance as proposed in [17,22]. For each of the 31 networks in the TSPLIB, four RSP benchmark problems are generated as follows. Let l_{ij} be the distance between nodes i and j in the TSP instance. To obtain optimal solutions visiting approximately 100%, 75%, 50%, and 25% of the total number of nodes in the instances, the coefficients in the objective function of the RSP are defined as: $c_{ij} = \lceil \alpha l_{ij} \rceil$, $[i,j] \in E$, $d_{ij} = \lceil (10 - \alpha) l_{ij} \rceil$, $(i,j) \in A$, for $\alpha = \{3,5,7,9\}$ and $p_i = 0$, $i \in V$. To illustrate the effect of considering each value of α in the optimal solution, Fig. 5 shows the optimal solution of problem KroA100 for the different values of α.

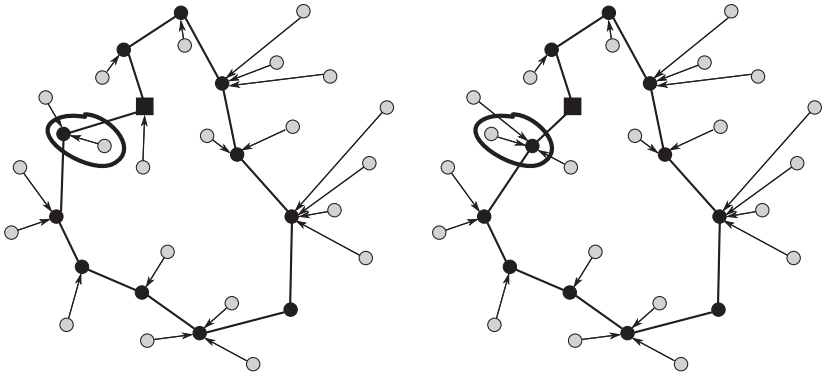
Finally, we have solved larger RSP instances which have been derived from the EUC-2D and EXPLICIT TSPLIB problems tsp225 to rat783 from the TSPLIB following the same idea.



(a) Cycle reduction



(b) Cycle augmentation



(c) Node exchange

Fig. 4. Local search procedures.

4.1. Selecting the configuration of the algorithm

The aim of the first part of the study was to assess the influence of five factors on the performance of the BBEA by defining a $4 \times 2^3 \times 3$ factorial design. The factors and levels are: Alpha ($\alpha = 3$, $\alpha = 5$, $\alpha = 7$, $\alpha = 9$), LS (local search = yes, local search = not), LKH (additional improvement = yes, additional improvement = not), population size ($IP = 400$, $IP = 800$) and subpopulation size ($PS = IP$, meaning that there are no subpopulations, $PS = 100$, $PS = 50$). Each combination of factors provides a configuration of the algorithm. Since we are going to assess the influence of the factor α in the achievement of the algorithm, instead of the above mentioned 124 benchmark problems, in this part of the study we take 31 test problems, each corresponding to a graph of the TSPLIB. Each of the test problems has been solved six times under each algorithm configuration. In order to facilitate the comparison of the results, the termination condition of the algorithm has been established in terms of computing time: 15 seconds if the number of nodes $n < 100$, 30 seconds if $100 \leqslant n < 150$ and 45 seconds if $150 \leqslant n$. Moreover, the migration interval MI has been fixed at 10 iterations.

Let f be the objective function value of the solution provided by the BBEA and $f_{best-known}$ be the objective function value of the best known solution in the literature for the corresponding problem. As we have previously mentioned, most benchmark problems have been solved optimally, but an upper bound only of the optimal objective function value is known for just a few problems. If $f - f_{best-known} \leqslant 0$, this is considered as success.

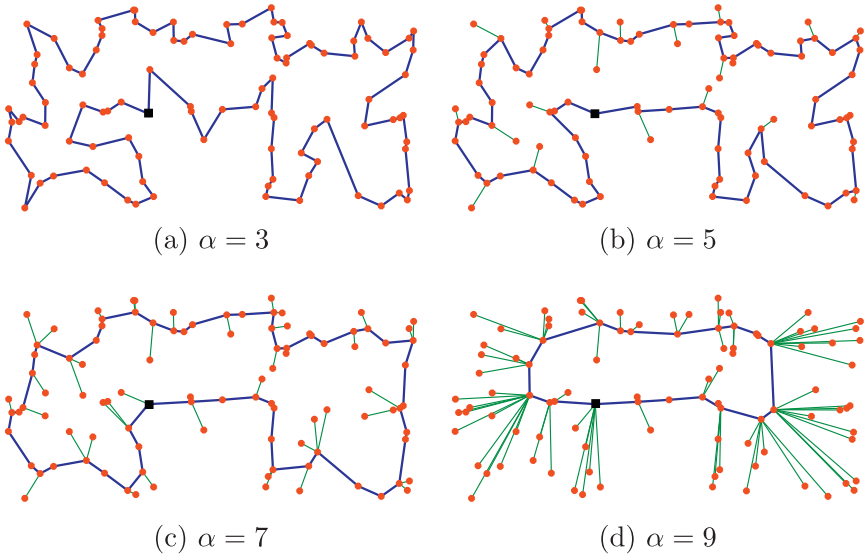


Fig. 5. Optimal solution of problem KroA100 for the different values of α.

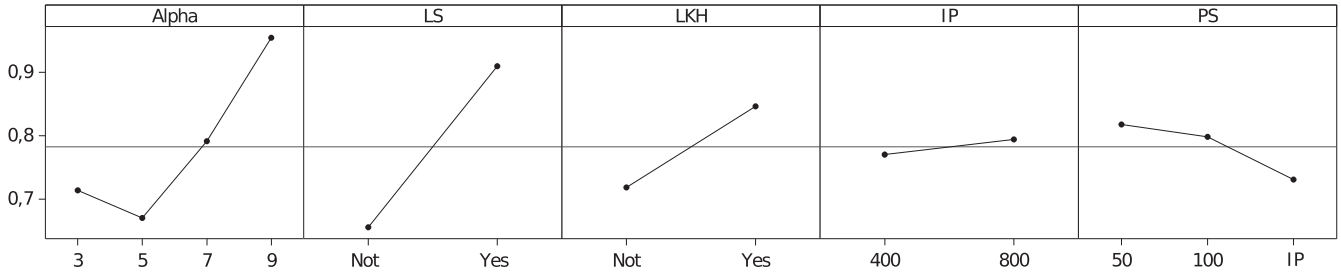


Fig. 6. Main effects plot for proportion of success.

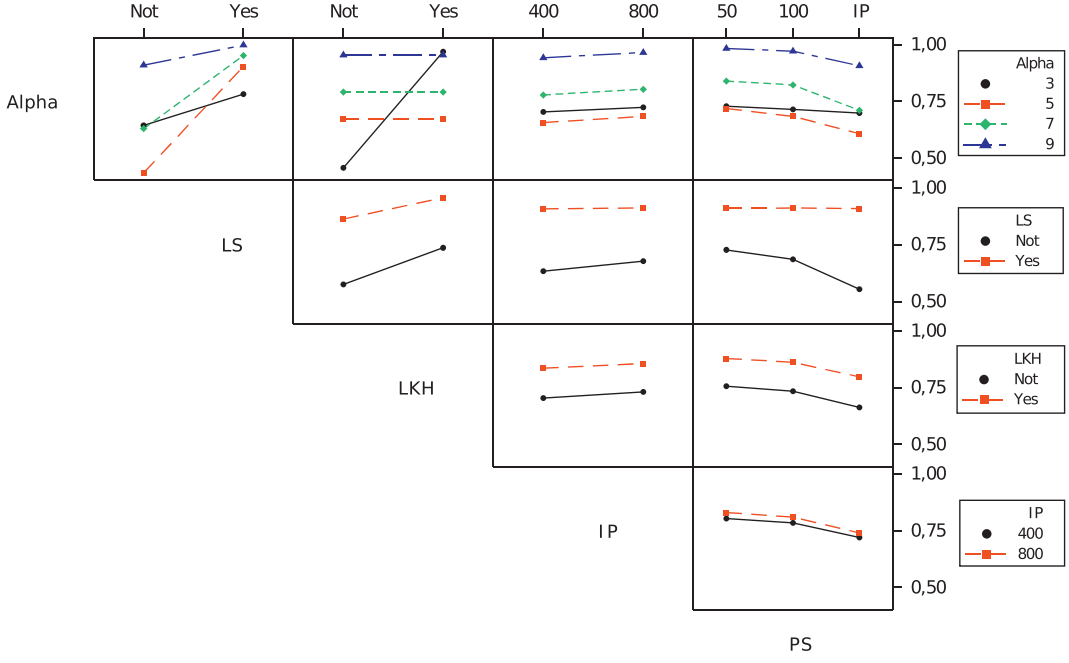


Fig. 7. Interaction plot for proportion of success.

The results of the Analysis of Variance (ANOVA) applied to the proportion of success indicate that the LS (29.4% of variability explained), Alpha (21.4% of variability explained) and LKH (7.5% of variability explained) factors are significant (the p-value being zero). The interactions between Alpha and the LKH (22.4% of variability explained), and the interaction between Alpha and LS (10.4% of variability explained) are also significant. Figs. 6 and 7 display the main factor plot and the interaction plot for the response variable proportion of success.

From this study we can conclude that, as we might expect, it is better to apply the local search. When doing this, the population and subpopulation sizes are not so relevant. Otherwise, it seems that a larger population size and a smaller subpopulation size perform better. In fact, if local search is applied, in 8119 out of the

Table 1
Results for benchmark problems.

Name	α	*Exact algorithm*		*BBEA*			
		Optimum solution	Corrected time (second)	Minimum	Maximum	% Error	Time (second)
eil51	3	1278 [17]	0.48	=	=		0.21
eil51	5	1995 [17]	0.24	=	=		1.64
eil51	7	2113 [17]	2.41	=	=		1.67
eil51	9	1244 [17]	2.41	=	=		1.88
berlin52	3	22,626 [17]	0.24	=	=		0.28
berlin52	5	36,115 [17]	0.96	=	=		1.86
berlin52	7	37,376 [17]	1.45	=	=		1.74
berlin52	9	20,361 [17]	2.89	=	=		1.91
brazil58	3	76,185 [17]	0.48	=	=		0.50
brazil58	5	115,045 [17]	3.37	=	=		2.26
brazil58	7	126,807 [17]	2.89	=	=		2.23
brazil58	9	83,690 [17]	6.75	=	=		2.28
st70	3	2025 [17]	3.13	=	=		0.36
st70	5	3110 [17]	5.78	=	=		2.61
st70	7	3402 [17]	4.82	=	=		2.35
st70	9	2610 [17]	16.87	=	=		2.72
eil76	3	1614 [17]	1.20	=	=		0.48
eil76	5	2460 [17]	5.06	=	=		3.06
eil76	7	2504 [17]	8.92	=	=		3.06
eil76	9	1710 [17]	35.66	=	=		3.57
pr76	3	324,477 [17]	255.18	=	=		0.60
pr76	5	500,395 [17]	35.42	=	=		3.22
pr76	7	555,845 [26]	20.47	=	=		3.02
pr76	9	424,359 [17]	44.34	=	=		3.19
rat99	3	3633 [17]	9.16	=	=		0.66
rat99	5	5885 [17]	8.43	=	=		4.23
rat99	7	6436 [17]	19.28	=	=		4.61
rat99	9	5150 [17]	105.78	=	=		4.86
kroA100	3	63,846 [17]	8.92	=	=		0.71
kroA100	5	100,785 [17]	20.72	=	=		4.23
kroA100	7	115,388 [17]	18.31	=	=		4.37
kroA100	9	94,265 [17]	86.02	=	=		5.40
kroB100	3	66,423 [17]	23.61	=	=		0.67
kroB100	5	104,550 [17]	13.73	=	=		4.44
kroB100	7	118,111 [17]	18.55	=	=		4.94
kroB100	9	93,938 [17]	109.40	=	=		5.72
kroC100	3	62,247 [17]	7.95	=	=		0.67
kroC100	5	99,065 [17]	10.84	=	=		4.48
kroC100	7	113,533 [17]	16.14	=	=		4.61
kroC100	9	92,894 [17]	102.41	=	=		5.33
kroD100	3	63,882 [17]	7.71	=	=		0.86
kroD100	5	101,645 [17]	9.16	=	=		5.60
kroD100	7	116,849 [17]	26.51	=	=		5.19
kroD100	9	92,102 [17]	100.72	=	=		5.52
kroE100	3	66,204 [17]	25.06	=	=		0.72
kroE100	5	104,915 [17]	27.47	=	=		5.25
kroE100	7	116,471 [17]	15.90	=	=		4.68
kroE100	9	96,116 [17]	121.69	=	=		5.94
rd100	3	23,730 [17]	3.86	=	=		0.69
rd100	5	37,975 [17]	24.10	=	=		4.96
rd100	7	40,915 [17]	27.23	=	=		5.10
rd100	9	31,776 [17]	101.45	=	=		5.22
eil101	3	1887 [17]	11.08	=	=		0.87
eil101	5	2905 [17]	7.47	=	=		4.54
eil101	7	2926 [17]	57.83	=	=		5.34
eil101	9	1955 [17]	120.00	=	=		5.45
lin105	3	43,137 [17]	5.06	=	=		0.74
lin105	5	69,365 [17]	10.36	=	=		4.44
lin105	7	83,597 [17]	23.61	=	=		4.95
lin105	9	69,920 [17]	108.43	=	=		5.86
pr107	3	132,909 [17]	5.54	=	=		1.29
pr107	5	210,465 [17]	12.05	=	=		4.99
pr107	7	259,571 [17]	26.99	=	=		5.30
pr107	9	264,918 [17]	97.59	=	=		5.35

8928 problems solved, i.e. 90.94%, we have obtained success, showing that irrespective of the algorithm configuration, the BBEA with local search provides very accurate solutions within short computing times. The level of Alpha is also relevant and is especially important in terms of deciding whether or not to apply the LKH additional improvement. In fact, only if $\alpha = 3$ is it worth applying this improvement. Hence, we propose to select for the algorithm a configuration in which local search is applied but the LKH improvement is only applied to problems with $\alpha = 3$.

4.2. Comparing the BBEA with the results provided in the literature

In the second part of the study the goal is to obtain evidence of the quality of the BBEA by comparing the results yielded by the algorithm with the best solutions provided in the literature as well as with the results provided by the RKEA [25]. For this purpose, bearing in mind the considerations of the previous section, we have chosen to apply local search with a probability of 0.5 and to apply the LKH additional improvement if $\alpha = 3$. Moreover, we have

Table 2
Results for benchmark problems.

Name	α	Exact algorithm		BBEA			
		Optimum solution	Corrected time (second)	Minimum	Maximum	% Error	Time (second)
gr120	3	20,826 [17]	14.70	=	=		1.14
gr120	5	31,480 [17]	38.07	=	=		18.73
gr120	7	32,301 [17]	47.95	=	=		7.57
gr120	9	24,322 [17]	167.23	=	=		7.70
pr124	3	177,090 [17]	181.20	=	=		0.95
pr124	5	286,115 [17]	57.35	=	=		6.17
pr124	7	358,853 [17]	49.88	=	=		6.33
pr124	9	340,153 [17]	232.53	=	=		7.19
bier127	3	354,846 [17]	35.42	=	=		1.49
bier127	5	539,955 [17]	51.08	=	=		10.01
bier127	7	567,110 [17]	103.86	=	=		10.31
bier127	9	347,845 [17]	533.01	=	=		8.94
ch130	3	18,330 [17]	30.60	=	=		1.27
ch130	5	28,790 [17]	38.07	=	=		8.46
ch130	7	32,707 [17]	164.82	=	=		9.35
ch130	9	23,639 [17]	317.11	=	=		8.73
pr136	3	290,316 [17]	217.83	=	=		1.36
pr136	5	468,520 [17]	34.46	468,610	468,660	0.023	22.17
pr136	7	491,981 [17]	86.99	=	=		9.27
pr136	9	387,327 [17]	369.40	=	=		8.62
pr144	3	175,611 [17]	37.35	=	175,770	0.075	1.94
pr144	5	290,945 [17]	737.59	291,030	291,030	0.029	10.09
pr144	7	383,041 [17]	106.51	=	=		10.44
pr144	9	366,833 [17]	372.05	=	=		9.78
ch150	3	19,584 [17]	73.98	=	=		1.41
ch150	5	31,170 [17]	75.90	=	=		9.30
ch150	7	34,930 [17]	135.18	=	=		10.57
ch150	9	26,371 [17]	783.61	=	=		11.25
kroA150	3	79,572 [17]	246.99	=	=		1.59
kroA150	5	125,435 [17]	53.98	=	125,440	0.001	16.02
kroA150	7	140,961 [17]	131.33	=	=		10.02
kroA150	9	113,080 [17]	659.28	=	=		11.23
kroB150	3	78,390 [17]	152.53	=	=		1.56
kroB150	5	122,875 [17]	1622.17	=	=		12.47
kroB150	7	135,382 [17]	101.45	=	=		9.49
kroB150	9	108,885 [17]	626.99	=	=		11.33
pr152	3	221,046 [17]	140.24	=	=		2.35
pr152	5	*364,990 [26]		364,605	364,820	−0.092	18.31
pr152	7	*467,024 [26]		=	468,027	0.056	11.56
pr152	9	475,440 [17]	755.42	=	=		12.57
u159	3	126,240 [17]	35.42	=	=		1.37
u159	5	204,250 [17]	132.05	=	=		10.20
u159	7	235,221 [17]	135.90	=	=		11.03
u159	9	199,552 [17]	865.06	=	199,573	0.002	17.07
rat195	3	6969 [17]	429.16	=	=		2.43
rat195	5	11,320 [17]	178.55	11,330	11,335	0.125	27.94
rat195	7	12,319 [17]	497.59	=	=		21.96
rat195	9	8977 [26]	787.50	=	=		18.18
d198	3	47,340 [17]	374.46	=	=		20.93
d198	5	76,945 [17]	1045.78	=	=		35.02
d198	7	94,300 [17]	1575.66	=	94,310	0.005	29.36
d198	9	96,088 [26]	4232.28	=	=		21.10
kroA200	3	*88,104 [26]		=	=		2.50
kroA200	5	*138,885 [26]		=	=		19.92
kroA200	7	158,227 [17]	1316.87	=	=		21.63
kroA200	9	122,594 [26]	285.22	=	=		20.48
kroB200	3	88,311 [17]	198.55	=	=		2.60
kroB200	5	138,905 [17]	372.05	=	=		20.41
kroB200	7	156,638 [17]	366.02	=	=		19.99
kroB200	9	124,043 [26]	1366.56	=	=		21.00

selected $IP = 800$ and $P = 16$, i.e. each subpopulation has a size $PS = 50$, and we have maintained the migration interval $MI = 10$ iterations. With this configuration, we have solved 6 times each of the 124 benchmark problems using a termination condition based on the number of successive iterations without improvement. Based on pilot testing, we have fixed this number at 5 iterations if $\alpha = 3$ and at 50 if $\alpha = 5, 7, 9$.

Moreover, due to the very different computational environments of the computational experiments provided in the literature, when displaying the results we have applied a correcting factor to the processing time in order to allow the comparison of computing times. This factor β is established as the ratio between the clocks of the machines on which the programs were run. It is worth pointing out that, although we had a multi-processor computer at hand, only one processor was used in our tests.

Tables 1 and 2 display the comparison with the best solution provided in the literature. The first and second columns show the names of the problems. The third column indicates the value of the optimum solution and the reference in which it is provided. Problems marked with an asterisk show the objective function value of the best solution known in the literature. The fourth column shows the computing time provided in the corresponding reference divided by the correcting factor β. This factor is $\beta = 4.15$ for Ref. [17] and $\beta = 1.27$ for Ref. [26]. The fifth and sixth columns display the minimum and the maximum of the objective function value of the solution obtained in the six runs of each problem. The = symbol means that the best result in the literature (third column) has been yielded by the BBEA. It is worth noting that in 115 out of the 124 problems (92.74%), the best known result in the literature has been matched in the six runs. Moreover, in the six runs of the

Table 3
Comparing the RKEA and the BBEA.

Name	α	*RKEA*			*BBEA*		
		% Error	Min. dev.	Corrected time (second)	% Error	Min. dev.	Time (second)
eil51	5	1.15		0.55			1.64
eil51	7			0.55			1.67
eil51	9			0.35			1.88
st70	5			2.65			2.61
st70	7			1.70			2.35
st70	9			1.90			2.72
eil76	5	0.08		3.10			3.06
eil76	7			2.35			3.06
eil76	9			2.30			3.57
pr76	5	0.09		2.95			3.22
pr76	7			4.75			3.02
pr76	9			2.90			3.19
rat99	5	1.10	0.76	13.55			4.23
rat99	7			9.25			4.61
rat99	9			9.75			4.86
kroA100	5	0.02		8.05			4.23
kroA100	7			8.45			4.37
kroA100	9			9.85			5.40
kroB100	5	0.06	0.02	7.70			4.44
kroB100	7	0.14		13.35			4.94
kroB100	9			7.95			5.72
kroC100	5	0.12		8.35			4.48
kroC100	7			7.03			4.61
kroC100	9			9.05			5.33
kroD100	5	0.45	0.29	9.30			5.60
kroD100	7	0.04		10.95			5.19
kroD100	9			10.75			5.52
kroE100	5	0.31	0.23	8.45			5.25
kroE100	7			8.15			4.68
kroE100	9			9.45			5.94
eil101	5	0.85	0.34	6.80			4.54
eil101	7			11.15			5.34
eil101	9			8.60			5.45
pr107	5	0.09		10.45			4.99
pr107	7	0.01		24.60			5.30
pr107	9			16.00			5.35
pr136	5	0.37	0.19	27.50	0.02	0.02	22.17
pr136	7	0.06		107.25			9.27
pr136	9			39.95			8.62
pr144	5	0.71	0.52	30.70	0.03	0.03	10.09
pr144	7			40.05			10.44
pr144	9			82.75			9.78
kroA150	5	0.54	0.15	57.70			16.02
kroA150	7	0.01		74.30			10.02
kroA150	9			52.70			11.23
kroB150	7			56.30			9.49
kroB150	9			58.80			11.33
pr152	9			109.20			12.57
rat195	5	2.60	2.25	137.35	0.13	0.09	27.94
rat195	7	0.39	0.32	327.35			21.96
kroB200	5	0.67	0.11	165.15			20.41
kroB200	7	0.05		261.05			19.99

problem pr152, $\alpha = 5$, the BBEA has provided a solution better than the best known solution. The seventh column shows the percentage of error defined as $100 \times (\bar{f} - f_{best-known})/f_{best-known}$, where $\bar{f}$ is the average of the objective function value of the solution yielded by the BBEA in the 6 runs of the problem. When the BBEA yields the best known solution in all runs, there is no error, hence the seventh column is empty. For the problem pr152, $\alpha = 5$ this error is negative, meaning that the BBEA provides a better solution. For the problems in which the best known result has not been achieved in all the six runs, the error ranges from 0.001% to 0.125%, which is a remarkably close fit. The average computing time in seconds in the six runs is shown in the eighth column. It ranges from 0.20 to 40.88 seconds, with an average of 7.17 seconds.

Table 3 presents the data that allow us to compare the two evolutionary algorithms, the RKEA and the BBEA. The first and second columns show the names of the problems. Note that Renaud et al. [25] restrict their tests to 52 benchmark instances which had been exactly solved by Labbé et al. [17], hence only these benchmark problems are included. The third and the sixth columns display the percentage of error as defined above. Notice that the RKEA results were obtained by successively running the algorithm five times, whereas the BBEA was run six times. The fourth and the seventh columns present the minimum deviation defined as $100 \times (f_{best} - f_{best-known})/f_{best-known}$, where f_{best} is the best value of the objective function yielded by the corresponding algorithm. When the corresponding algorithm yields the optimum solution in all runs, there is no error, hence the %Error column is empty. Similarly if the algorithm reaches the optimum solution in any run, the minimum deviation column is empty. Finally, the fifth and the eighth columns show the average computing times for the algorithms RKEA and BBEA, respectively. Original computing times for the RKEA have been divided by the correcting factor $\beta = 4$. It should be noted that the RKEA was stopped after 5 successive iterations without improvement, whereas for the BBEA the number of successive iterations without improvement was established at 50 if $\alpha = 5, 7, 9$.

Summarizing data in Table 3, in 49 out of 52 instances, the BBEA reaches the optimum solution in all the six runs. The RKEA provides the optimum in all the five runs in 29 out of the 52 instances and reaches the optimum in at least one of the five runs in 41 out of the 52 instances. Moreover, the average minimum deviation from the optimum is 0.0026 and 0.0996, respectively for the BBEA and the RKEA. Also, the average of the percentage of error is 0.0034 against 0.1906. Finally, concerning the computing times, the average (per run) computing time is 7.378 against 36.560. Hence, we can conclude that the BBEA outperforms the RKEA.

4.3. Solving larger problems

Taking into account the efficiency of the BBEA when applied to solving instances of the RSP with a number of nodes between 50 and 200, we have considered it useful to study its performance in larger problems. These RSP instances have been derived from the EUC-2D and EXPLICIT TSPLIB problems tsp225 to rat783 following the same idea as in [17,22]. We have used the same algorithm configuration and the same termination condition described in the previous subsection. Again, each problem has been solved six times.

The results are presented in Table 4. The first and second columns indicate the names of the problem. The third column indicates the objective function value of the best solution obtained in the six runs of each problem. Similarly, the fourth column displays the maximum of the objective function value of the solution provided by the BBEA in the six runs of each problem. The = symbol means that both values are equal. The fifth column shows the per-

Table 4
Results for larger benchmark problems.

Name	α	Min	Max	% Error	Time (second)
tsp225	3	11,748	=		4.33
tsp225	5	19,200	=		30.67
tsp225	7	20,413	=		39.83
tsp225	9	15,806	=		25.17
pr226	3	241,107	=		7.00
pr226	5	383,055	=		19.50
pr226	7	469,493	=		20.50
pr226	9	470,711	=		21.50
gil262	3	7134	=		8.33
gil262	5	11,235	=		34.17
gil262	7	12,497	=		40.00
gil262	9	9749	=		44.33
pr264	3	147,405	=		7.67
pr264	5	238,135	238,190	0.012	40.83
pr264	7	285,545	=		64.50
pr264	9	256,842	=		39.33
a280	3	7737	=		9.33
a280	5	12,655	=		31.83
a280	7	14,109	=		57.83
a280	9	10,742	=		39.67
pr299	3	144,573	=		11.83
pr299	5	232,065	232,070	0.000	72.50
pr299	7	260,209	=		56.67
pr299	9	213,804	=		56.67
lin318	3	126,087	=		12.33
lin318	5	202,140	=		57.33
lin318	7	229,449	=		51.67
lin318	9	177,089	=		52.83
rd400	3	45,843	=		16.67
rd400	5	72,565	72,590	0.017	239.50
rd400	7	79,960	=		117.50
rd400	9	59,496	=		117.00
fl417	3	35,583	=		236.00
fl417	5	56,880	56,900	0.007	141.33
fl417	7	70,983	70,987	0.004	160.33
fl417	9	72,767	=		141.17
pr439	3	321,651	=		29.50
pr439	5	514,700	514,905	0.013	320.00
pr439	7	609,582	609,878	0.032	263.33
pr439	9	473,540	473,659	0.008	178.33
pcb442	3	152,334	=		19.50
pcb442	5	242,105	242,195	0.014	225.17
pcb442	7	266,359	266,499	0.030	222.83
pcb442	9	195,736	195,783	0.005	141.33
d493	3	105,006	=		45.50
d493	5	166,880	166,950	0.025	306.17
d493	7	185,364	185,569	0.041	399.17
d493	9	151,786	151,801	0.007	216.67
si535	3	145,359	=		79.17
si535	5	241,800	241,825	0.007	289.17
si535	7	194,976	=		190.33
si535	9	90,693	=		303.17
pa561	3	8289	=		45.67
pa561	5	12,825	12,850	0.104	552.17
pa561	7	13,331	13,339	0.031	491.33
pa561	9	8952	9027	0.320	451.50
u574	3	110,715	=		35.50
u574	5	175,475	175,795	0.088	391.50
u574	7	192,895	=		307.50
u574	9	146,604	146,660	0.012	407.83
rat575	3	20,322	=		28.17
rat575	5	32,400	32,435	0.054	379.17
rat575	7	34,734	34,770	0.039	459.17
rat575	9	24,770	=		344.00
p654	3	103,929	=		410.33
p654	5	167,230	167,305	0.013	346.83
p654	7	195,797	195,801	0.000	729.33
p654	9	197,333	197,353	0.002	366.50
d657	3	146,739	=		51.17
d657	5	231,660	=		696.00
d657	7	263,072	=		858.17
d657	9	207,622	=		512.33
u724	3	125,730	=		57.50
u724	5	200,555	200,880	0.095	1281.50

Table 4 (*continued*)

Name	α	Min	Max	% Error	Time (second)
u724	7	220,301	220,709	0.066	1237.50
u724	9	167,025	167,266	0.047	798.00
rat783	3	26,418	=		54.17
rat783	5	42,490	42,645	0.143	1346.17
rat783	7	46,789	46,841	0.056	1156.33
rat783	9	33,636	33,660	0.024	1135.50

centage of error computed as $100 \times (\bar{f} - f_{min})/f_{min}$, where f_{min} is the value contained in column three and $\bar{f}$ is the average of the objective function values of the six solutions. Finally, the last column displays the average computing time involved in seconds.

Notice that for most of the smaller problems, the BBEA provides the same solution in all the six runs. When the algorithm does not provide in all the six runs the same solution, the percentage of error ranges from 0.000 to 0.320, thus confirming the accuracy of the algorithm. Moreover, the computing time ranges from 4.33 seconds to less than 10 minutes for most problems. Only some problems with more than 650 nodes need up to 23 minutes to provide a solution. Confirming the results obtained in the previous subsection, we have also observed that applying the additional improvement LKH is only of interest for $\alpha = 3$. For problems with $\alpha = 5,7,9$ there are no differences whether the LKH is applied or not except for $\alpha = 5$ and the problems si535, rat575 and rat783 in which there are slight differences. In problem si535-5, the minimum computed is 241,700, in problem rat575-5 it is 32,395 and in problem rat783-5 it is 42,485.

5. Conclusions

A fast and efficient evolutionary algorithm for the RSP has been presented in this paper. It is based on a new formulation of this problem as a binary bilevel programming problem with one leader and two independent followers. The algorithm is based on an encoding scheme which handles the nodes on the ring. A computational study has allowed us to conclude that applying local search after the usual crossover and mutation operators vastly improves the performance of the algorithm. Also, the computational results show that the algorithm outperforms the heuristic algorithms proposed in the literature in terms of the number of benchmark problems which are solved optimally. In fact, in 92.74% of benchmark problems, the best known result in the literature has been matched in the six runs. Moreover, for one of the test problems, the algorithm has provided a solution better than the best known solution in all the six runs. The computing time can be considered very short (it ranges from 0.20 to 40.88 seconds, with an average of 7.17 seconds). In fact the computing times are much lower than those reported in the literature, even when we take into account the correcting factor due to the different characteristics of the computers used. The algorithm has also been tested on a set of larger benchmark problems (up to 783 nodes) and the computational results are very satisfactory in terms of accuracy and computing time. Future work involves the application of the underlying ideas of the problem reformulation proposed in this paper to other combinatorial problems. Sharing the decision process can help to improve the computing times in solving the problems.

Acknowledgments

The authors gratefully acknowledge the anonymous referees for their valuable suggestions to improve the presentation of the paper.

References

[1] M. Affenzeller, S. Wagner, S. Winkler, A. Beham, Genetic algorithms and genetic programming: modern concepts and practical applications, Chapman & Hall/CRC, 2009.
[2] R. Baldacci, M. Dell'Amico, Heuristic algorithms for the multi-depot ring star problem, European Journal of Operational Research 203 (1) (2010) 270–281.
[3] R. Baldacci, M. Dell'Amico, J.J. Salazar González, The capacitated m-ring star problem, Operations Research 55 (6) (2007) 1147–1162.
[4] J.F. Bard, Practical Bilevel Optimization, Algorithms and Applications, Kluwer Academic Publishers, Dordrecht, Boston, London, 1998.
[5] J. Bean, Genetic algorithms and random keys for sequencing and optimization, Journal of Computing 6 (1994) 154–160.
[6] H.I. Calvete, C. Galé, Linear bilevel multi-follower programming with independent followers, Journal of Global Optimization 39 (3) (2007) 409–417.
[7] H.I. Calvete, C. Galé, Linear bilevel programs with multiple objectives at the upper level, Journal of Computational and Applied Mathematics 234 (4) (2010) 950–959.
[8] H.I. Calvete, C. Galé, Modeling and solving linear bilevel problems with multiple objectives at the lower level, Omega - The International Journal of Management Science 39 (1) (2011) 33–40.
[9] R. Chion, T. Weise, Z. Michalewicz (Eds.), Variants of Evolutionary Algorithms for Rear-World Applications, Springer, Berlin, 2012.
[10] B. Colson, P. Marcotte, G. Savard, An overview of bilevel programming, Annals of Operations Research 153 (2007) 235–256.
[11] S. Dempe, Foundations of Bilevel Programming, Kluwer Academic Publishers., Dordrecht, Boston, London, 2002.
[12] T.C.S. Dias, G.F. de Sousa Filho, E.M. Macambira, L.A.F. Cabral, M.H.C. Fampa, An efficient heuristic for the ring star problem, in: C. Alvarez, M. Serna (Eds.), Experimental Algorithms, Lecture Notes in Computer Science, Springer Verlag, 2006, pp. 24–35 (No. 4007).
[13] K. Helsgaun, Effective implementation of the Lin–Kerninghan traveling salesman heuristic, European Journal of Operational Research 126 (2000) 106–130.
[14] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[15] D.S. Johnson, L.A. McGeoch, Experimental analysis of heuristics for the stsp, in: G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and Its Variations, Kluwer Academic Publishers, Dordrecht, 2002, pp. 369–443.
[16] S. Kedad-Sidhoum, V.H. Nguyen, An exact algorithm for solving the ring star problem, Optimization 59 (1) (2010) 125–140.
[17] M. Labbé, G. Laporte, I. Rodríguez Martín, J.J. Salazar González, The ring star problem: Polyhedral analysis and exact algorithm, Networks 43 (3) (2004) 177–189.
[18] G. Laporte, A concise guide to the traveling salesman problem, Journal of the Operational Research Society 51 (2010) 35–40.
[19] A. Liefooghe, L. Jourdan, E.G. Talbi, Metaheuristics and cooperative approaches for the bi-objective ring star problem, Computers and Operations Research 37 (6) (2010) 1033–1044.
[20] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Operations Research 21 (2) (1973) 498–516.
[21] Z. Michalewick, Genetic Algorithms + Data Structures = Evolution Programs, third ed., Springer, Berlin, 1996.
[22] J.A. Moreno Pérez, J.M. Moreno Vega, I. Rodríguez Martín, Variable neighborhood tabu search and its application to the median cycle problem, European Journal of Operational Research 151 (2) (2003) 365–378.
[23] Z. Naji-Azimi, M. Salari, P. Toth, An integer linear programming based heuristic for the capacitated m-ring-star problem, European Journal of Operational Research 217 (1) (2012) 17–25.
[24] G. Reinelt, Tsplib – a traveling salesman problem library, Journal of Computing 3 (1991) 376–384.
[25] J. Renaud, F.F. Boctor, G. Laporte, Efficient heuristics for median cycle problems, Journal of the Operational Research Society 55 (2) (2004) 179–186.
[26] L. Simonetti, Y. Frota, C.C. de Souza, The ring-star problem: a new integer programming formulation and a branch-and-cut algorithm, Discrete Applied Mathematics 159 (16) (2011) 1901–1914.
[27] R. Tanese, Distributed genetic algorithms, in: J.D. Schaffer (Ed.), Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann, 1989, pp. 434–439.