Computational and Applied Mathematics (2019) 38:188
https://doi.org/10.1007/s40314-019-0976-4

The multicommodity traveling salesman problem with priority prizes: a mathematical model and metaheuristics

Tiago Tiburcio da Silva[1] · Antônio Augusto Chaves[1] · Horacio Hideki Yanasse[1] · Henrique Pacca Loureiro Luna[2]

Received: 4 February 2019 / Revised: 24 May 2019 / Accepted: 29 September 2019 /
Published online: 10 October 2019

Abstract
The classic Traveling Salesman Problem (TSP) only considers the costs involved in the routes and does not differentiate products or customers. Logistic companies face conflict between operational costs, customers with different categories of products, and customer satisfaction, which is directly related to delivery time. This paper presents a new mathematical model for a TSP with variable costs and priority prizes, taking into account the customer's product and preference values. This problem is denoted as the Multicommodity Traveling Salesman Problem with Priority Prizes (MTSPPP). Two versions of the Biased Random-Key Genetic Algorithm (BRKGA) are proposed to solve medium and large instances of the MTSPPP. Computational tests were performed, using modified instances based on classical TSP instances. The proposed methods have proved to be efficient in solving the MTSPPP.

Keywords Traveling salesman · Multicommodity · Priority · Genetic algorithm

Mathematics Subject Classification 90-08

1 Introduction

The Traveling Salesman Problem (TSP) is a well-known NP-hard problem in network optimization (Applegate et al. 2006; Dantzig et al. 1954; Flood 1956; Lawler et al. 1985). It consists of determining a minimum cost Hamiltonian path, visiting all customers only once,

Communicated by Hector Cancela.

Tiago Tiburcio da Silva
ttsilva@gmail.com

Antônio Augusto Chaves
antonio.chaves@unifesp.br

Horacio Hideki Yanasse
horacio.yanasse@unifesp.br

Henrique Pacca Loureiro Luna
pacca@ic.ufal.br

1 Univ. Fed. of São Paulo, São José dos Campos, SP 12231-280, Brazil

2 Federal University of Alagoas, Maceió, AL 57072-970, Brazil

and returning to the starting point. The TSP arises in many applied settings, such as drilling of printed circuit boards (Grotschel et al. 1991), the order-picking problem in depots (Ratliff and Rosenthal 1983), the school bus routing problem (Angel et al. 1972), the defender–attacker–defender problem (Lozano et al. 2017), and the maritime logistics (Malaguti et al. 2018).

The TSP and most of its variations are oriented by costs and, in the literature, we hardly see studies considering different demands. One variation that does consider different demands is the Multicommodity Traveling Salesman Problem (MTSP) (Sarubbi 2003; Sarubbi and Luna 2007; Sarubbi et al. 2007). In the MTSP, product types and quantities that pass through a route connecting two customers are considered in the total cost. In practice, customers with larger demands or more precious or high-risk products must be delivered with a higher priority. For example, sensitive materials may require special transport structure, perishable products must be refrigerated, both leading to higher transportation costs. The authors (Sarubbi 2003) consider variable costs for each product type in each route between two customers and propose a network flow model that is solved by a Lagrangian relaxation and a branch-and-cut method (Sarubbi 2008; Sarubbi et al. 2008).

Another variation of the TSP considers priority prizes for customers along the route. These prizes are collected by the salesman according to the order each customer is visited. In this variation, the quality of customer service and delivery priorities are considered because customer preferences in terms of delivery sequence order must be quantified and optimized. Pureza et al. (2018) modeled this problem as a special case of a quadratic assignment problem (Koopmans and Beckmann 1957) called the Travelling Salesman Problem with Priority Prizes (TSPPP) and they solved it by a Tabu Search metaheuristic (Glover 1986). Silva (2017); Silva et al. (2018) used the TSPPP and the Profitable Tour Problem (PTP) (Archetti et al. 2014; Balas 1987) to formulate the problem of elaborating touristic itineraries considering variables such as the profile of the visitors, the planning of the trip, visitors' preferences and touristic attractions. The authors considered a case study in the City of Belém, in the State of Pará, Brazil, solving the problem by a Tabu Search metaheuristic.

Besides these situations, logistic companies face conflict between operational costs, customers with different categories of products, and customer satisfaction, which is directly related to delivery time. It then becomes a challenge to choose between minimizing travel costs and ensuring a degree of service quality for all customers. To obtain solutions that balance operating costs and customer satisfaction, we propose a new model that combines the TSPPP with MTSP.

In this paper, we look at the TSP from both the customer's and the salesman's points of view. We consider the objective of minimizing total costs, while satisfying customers' preference, by maximizing the priority prizes. Our model is based on the assignment problem and a network flow problem. Commodities flowing in the network represent the products. We denote this problem as Multicommodity Traveling Salesman Problem with Priority Prizes (MTSPPP).

In this study, we used two versions of the Biased Random-Key Genetic Algorithm (BRKGA) (Gonçalves and Resende 2011) to solve medium and large instances of the MTSPPP. A local search based on Iterated Local Search (ILS) (Lourenço et al. 2003) and Variable Neighborhood Descent (VND) (Hansen et al. 2010) were also used to improve the solutions found by BRKGA.

We generated some instances for the MTSPPP based on classical instances of TSP and we performed computational tests with the model and the metaheuristics. The model was able to solve small instances and the proposed metaheuristics efficiently solved medium and large instances. To the best of our knowledge, the MTSPPP has not been addressed before in the literature.

This article is organized as follows. A mathematical model of MTSPPP is presented in Sect. 2. In Sect. 3, the BRKGA+CS and A-BRKGA+CS algorithms are briefly introduced and in Sect. 4, a framework of BRKGA+CS and A-BRKGA+CS are presented in detail to solve the MTSPPP. The computational results for the proposed model and methods are presented and analyzed in Sect. 5. Finally, in Sect. 6, some conclusions are presented.

2 The multicommodity traveling salesman problem with priority prizes

Let N be the number of customers to be visited including the depot. Consider a graph $G(V, A)$, where V is a set composed by N nodes, numbered 1 to N, and A is a set of arcs. For convenience, node 1 is the depot and the other nodes represent the customers to be visited. The set of arcs A in this graph represent the paths between customers. For simplicity, we consider that product l, a commodity, will be delivered to customer l. We also assume that each customer can order only one product type, i.e., the same node (customer) cannot order different products. Denote c_{ij} as the fixed cost in the arc (i, j), q_l as the demand required by customer l, d_{ijl} as the variable cost to pass product l through arc (i,j), and p_{ki} the prize value collected when customer i is visited in the kth order. We assume, without loss of generality, that $q_l > 0$ and integer for all l.

Let x_{ki} be a binary decision variable equal to 1, if customer i is visited in the kth order, and 0, otherwise. Similarly, let y_{ij} be a binary decision variable equal to 1, if customer j is visited immediately after customer i, and 0, otherwise. Let f_{ijl} be a real decision variable corresponding to the flow quantity of the product l that is transported from customer i to customer j. Observe that in an optimal solution, due to constraints (7), (8), and (9) of the model, the values of the f_{ijl} variables will be integer if q_l is integer for all l. So, a mathematical formulation for the problem MTSPPP is the following:

$$\text{Max } \ Z = \sum_{k=2}^{N}\sum_{i=2}^{N} p_{ki}x_{ki} - \sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N}\left(c_{ij}y_{ij} + \sum_{l=2}^{N} d_{ijl}f_{ijl}\right) \tag{1}$$

subject to:

$$\sum_{i=1}^{N} x_{ki} = 1, \quad k = 1, 2, \ldots, N \tag{2}$$

$$\sum_{k=1}^{N} x_{ki} = 1, \quad i = 1, 2, \ldots, N \tag{3}$$

$$\sum_{j=1}^{N} y_{ij} = 1, \quad i = 1, 2, \ldots, N \tag{4}$$

$$\sum_{h=1}^{N} y_{hi} = 1, \quad i = 1, 2, \ldots, N \tag{5}$$

$$x_{(k-1)h} + x_{ki} - y_{hi} \leq 1, \quad h \neq i = 1, 2, \ldots, N, \ k = 2, 3, \ldots, N \tag{6}$$

$$\sum_{j=2}^{N} f_{1jl} = q_l, \quad l = 2, 3, \ldots, N \tag{7}$$

$$\sum_{\substack{i=1\\ i\neq l}}^{N} f_{ill} = q_l, \quad l = 2, 3, \ldots, N \tag{8}$$

$$\sum_{\substack{h=1\\ i\neq l, h\neq i}}^{N} f_{hil} - \sum_{\substack{j=1\\ i\neq l, i\neq j}}^{N} f_{ijl} = 0, \quad l, i = 2, 3, \ldots, N \tag{9}$$

$$f_{ijl} \leq q_l y_{ij}, \quad i \neq j = 1, 2, \ldots, N, \ l = 2, 3, \ldots, N \tag{10}$$

$$f_{ijl} \geq 0, \quad i, j = 1, 2, \ldots, N, \ l = 2, 3, \ldots, N, \tag{11}$$

$$x_{ki}, y_{ij} \in \{0, 1\}, \quad k, i, j = 1, 2, \ldots, N, i \neq j. \tag{12}$$

The objective function (1) maximizes the priority prizes while it minimizes the fixed total costs and variable costs. Constraints (2) and (3) impose the constraint that each customer must be visited only once. Constraints (4) and (5) are assignment constraints. Constraints (6) links the variables x and y. It ensures that if customer h is visited in the $(k-1)$th position, and customer i is visited in the kth position, then the arc (h,i) will be used.

Constraints (7) ensures that all products will leave the depot with their respective demands, and constraints (8) guarantees that all products will reach their destinations, taking into account the demand required. Constraints (9) enforces the conservation of flow at any node that is not the final destination for the products. Constraints (10) links the variables f and y. It ensures that no flow is allowed on an arc (i, j) unless it is used. Finally, constraints (11) and (12) define the domain of the decision variables. We set the first position in the route at 1, i.e., $x_{11} = 1$, since, for convenience, we set node 1 as the depot.

The subtour elimination constraints are implied by constraints (4), (5), (7)–(12). Node 1 is the supply node for all products $l, l = 2, 3, \ldots, N$. Therefore, in a feasible solution, there must be a path from node 1 to node $l, l = 2, 3, \ldots, N$, where the quantity q_l of product l has to flow to satisfy the demand of node $l, l = 2, 3, \ldots, N$, respectively. At each node in this path, we must have the conservation of flow, imposed by constraints (9). According to (10) and (12), the corresponding y_{ij} of the arcs in these paths must have value 1. Assume there is a subtour in the solution. There are two possibilities: (i) node 1 does not belong to this subtour; (ii) node 1 belongs to this subtour. Case (i) cannot happen for there is no path from node 1 to any of the nodes belonging to the subtour. Therefore, the demands of these nodes cannot be satisfied. In case (ii), there is at least one node, say t, that does not belong to the subtour. Since there must be a path from node 1 to node t to satisfy the demand of this node, then we have two possibilities: this path from node 1 to node t contains only nodes not belonging to the subtour or, this path contains other nodes of the subtour before reaching node t. The first case cannot happen because of constraint (4) that imposes that there is only a single arc with flow leaving node 1. Hence, we cannot have two arcs with positive flows leaving node 1, one to supply the demand of node t and the other to supply the demands of the nodes belonging to the subtour. The second case also cannot happen because of constraint (4). Let us consider the node, say s, belonging to the subtour, where the flow to node t leaves the subtour. Again, we cannot have two arcs with positive flows leaving node s, one to supply the demand of node t and the other to supply the demands of the nodes belonging to the subtour.

Computational experiments were performed with this model, using CPLEX 12.6.3 and are presented in Sect. 5. The model was able to solve instances with at most 28 customers in 10800 seconds. For medium and large instances, the solver was not able to find a feasible solution within the time limit set.

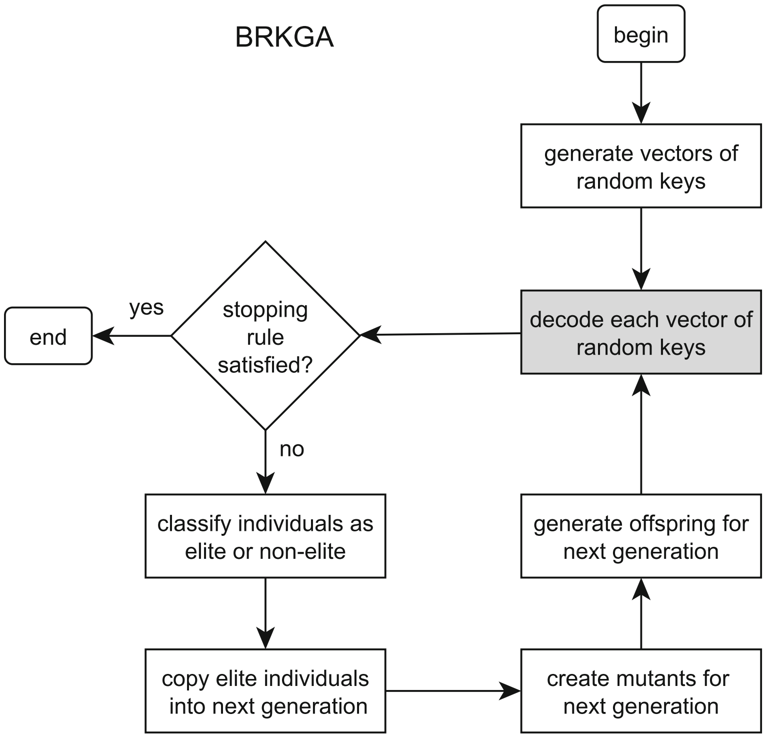
BRKGA
begin
generate vectors of random keys
decode each vector of random keys
stopping rule satisfied?
yes
end
no
classify individuals as elite or non-elite
copy elite individuals into next generation
create mutants for next generation
generate offspring for next generation

Fig. 1 The BRKGA framework. Adapted from Chaves et al. (2016, 2018)

3 BRKGA

The Biased Random-Key Genetic Algorithm (BRKGA) was proposed by Gonçalves and Resende (2011). The method consists of a specialization of the RKGA (Random-Key Genetic Algorithm) (Bean 1994), an algorithm that evolves a population of random keys, where each vector of n random keys represents a solution of a combinatorial optimization problem. A random key is a real number randomly generated in the continuous interval [0, 1). A vector of random keys is decoded into a feasible solution and an objective value (or *fitness*) through a deterministic algorithm called a *decoder*, which is dependent on the problem to be solved.

A population of p individuals is made up. At each generation, the population is divided into two sets, according to the fitness of each individual. The p_e best fitness vectors are identified as an elite set and the rest of the population as a non-elite set. All elite vectors are copied, without change, to the next generation and p_m mutants are introduced. A mutant is a vector of random keys generated in the same way as all individuals of the initial population. The $p - p_\mathrm{e} - p_\mathrm{m}$ remaining individuals are generated by combining pairs of randomly chosen solutions an elite set and another non-elite set (Spears and Jong 1991).

In Fig. 1, we illustrate a flow diagram of the BRKGA. In this diagram, the gray rectangle represents the decoder function. This is the single component of a BRKGA that depends on the problem being solved.

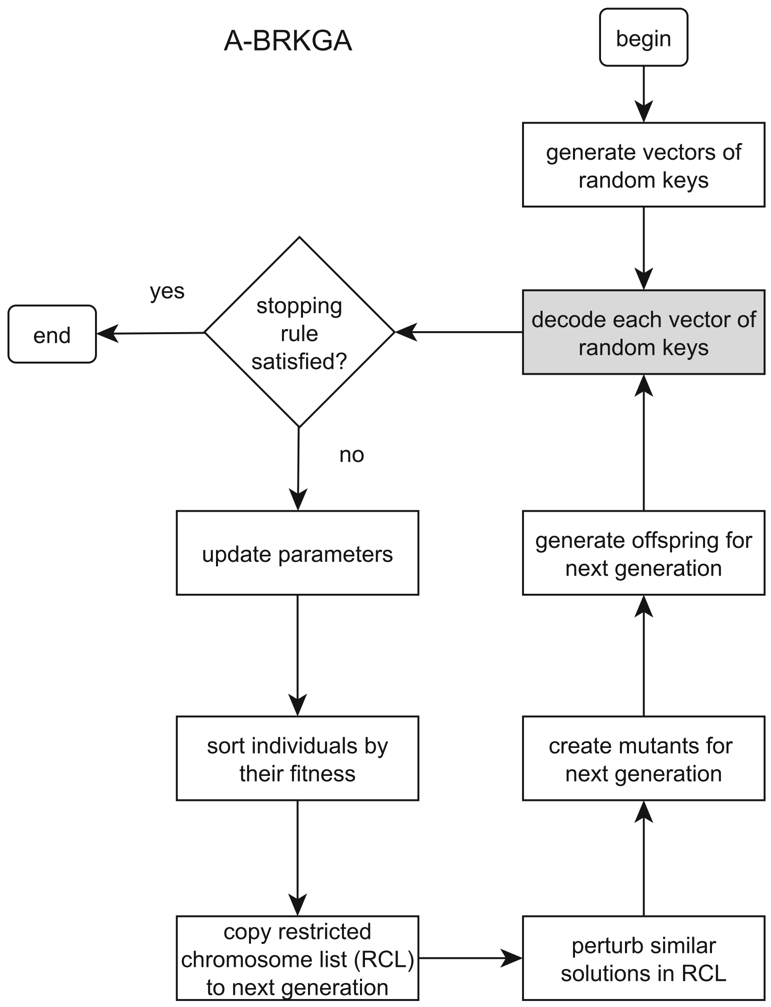
A-BRKGA
begin
generate vectors of random keys
decode each vector of random keys
stopping rule satisfied?
yes
end
no
update parameters
sort individuals by their fitness
copy restricted chromosome list (RCL) to next generation
perturb similar solutions in RCL
create mutants for next generation
generate offspring for next generation

Fig. 2 The Adaptive BRKGA framework. Adapted from Chaves et al. (2016, 2018)

3.1 Adaptive BRKGA (A-BRKGA)

The Adaptive Biased Random-Key Genetic Algorithm (A-BRKGA) is a recent metaheuristic proposed by Chaves et al. (2018). This metaheuristic tunes parameters on-line so that the processes of intensification and diversification are balanced. In Fig. 2, we illustrate the evolutionary process of the A-BRKGA.

In A-BRKGA, a solution of a combinatorial optimization problem is also represented by a vector of n random-keys plus two extra random-keys in positions $n + 1$ and $n + 2$. They are used to compute the self-adaptive parameters.

If we denote p_{max} and p_{min} as the maximum and the minimum sizes of the population, respectively, the maximum number of generations (max_{gen}), the stop criterion, is given by the equation:

$$max_{gen} = \gamma^{\left(\log_{\gamma}^{p_{min}} - p_{max}\right)}. \tag{13}$$

Equation (13) is motivated by the expression used in Simulated Annealing (Ingber 1993) to calculate the maximum number of required iterations for a given temperature. The parameters p_{max} and p_{min} are set at 1000 and 100, respectively. These values are recommended in the literature (Gonçalves and Resende 2016; Prasetyo et al. 2015). The parameter γ is selected from the set $\{0.999, 0.998, 0.997\}$ and depends on the available running time.

At each generation, the population is divided into two groups: the first group (RCL, restricted chromosome list) consists of the best individuals evaluated by *fitness* and the second group composed of the individuals of the population not in RCL. The RCL is composed of individuals whose *fitness* is less than a bound f_e (minimization problem), given by a convex combination of the *fitness* values of the best (f_{min}) and the worst (f_{max}) individuals in the current population:

$$f_\text{e} = \alpha f_{max} + (1 - \alpha) f_{min}, \tag{14}$$

with $\alpha \in [0, 1]$. The number of individuals in the RCL is limited by $p_\text{e} = p * \kappa_e$, where κ_e is given by Eq. (15). In Eq. (15), g_k is the number of the current generation. According to the literature, the range of the number of elite individuals should vary between 10 and 25% of the population (Gonçalves and Resende 2016; Prasetyo et al. 2015). The number of elite individuals is lower in the initial generations when the average quality of the individuals is low and this number increases in the subsequent generations.

$$\kappa_e = 0.10 + \frac{g_k}{max_{gen}} * (0.25 - 0.10). \tag{15}$$

The parameter α in (14) is modified at each iteration according to Eq. (16). It starts with the value 0.3 and ends up with the value 0.1. f_e decreases during the evolution of the population because of the tendency of flatness of the individuals fitness after a certain number of generations.

$$\alpha = 0.10 + \left(1 - \frac{g_k}{max_{gen}}\right) * (0.30 - 0.10). \tag{16}$$

The individuals in the RCL with the same *fitness* are perturbed by an adaptive parameter β, that is the probability of randomly swapping two random key values, and that is updated according to Eq. (17), where x_{n+2} is the random-key in position $n + 2$ of the individual x in the current generation. This perturbation maintains the diversity of the population and avoids premature convergence.

$$\beta = 0.001 + x_{n+2} * (0.10 - 0.001). \tag{17}$$

All RCL individuals in the population of generation k are copied to the population of generation $k + 1$. At each generation $p_\text{m} = p * \kappa_m$ individuals are generated as in the initial population and inserted to the population of generation $k + 1$, where κ_m is given by Eq. (18). These individuals are named mutants. Based on the literature, the number of mutants should range between 5 and 20% of the population (Gonçalves and Resende 2016; Prasetyo et al. 2015). As the population evolves, the number of mutants decreases aiming at a greater intensification in the search for the solution.

$$\kappa_m = 0.05 + \left(1 - \frac{g_k}{max_{gen}}\right) * (0.20 - 0.05). \tag{18}$$

To complete the population, $p - |\text{RCL}| - p_\text{m}$ additional individuals are needed. These individuals are obtained by combining pairs of individuals of the current population, using the *parameterized uniform crossover* (Spears and Jong 1991). A number $r_j \in [0, 1]$ is generated for each individual gene. If $r_j \leq \rho_e$, then this individual gene inherits the allele of the RCL parent. Otherwise, it inherits the allele of the other parent. The parameter ρ_e is self-adaptive and is given by equation:

$$\rho_e = 0.65 + y_{n+1} * (0.80 - 0.65), \tag{19}$$

where y_{n+1} is the random-key in position $n + 1$ of the individual y that is not in RCL. The probability of accepting an elite allele is always between 65 and 80% as recommended in the literature (Chaves et al. 2018).

All ranges of the parameters used in the A-BRKGA were based on studies in the literature and on empirical tests. They were defined aiming the development of an algorithm that easy reuse code and is efficient to solve different combinatorial optimization problems. In Sect. 3.2, we present a local search heuristic used with BRKGA and A-BRKGA.

3.2 Local search

Chaves et al. (2018) showed that the use of local search heuristics can improve the performance of BRKGA and A-BRKGA. However, to avoid a significant increase in running time these heuristics are applied only to regions considered promising by the algorithm. To find these promising regions, the Label Propagation (LP) method (Raghavan et al. 2007) is used to identify communities with a great number of similar individuals within the RCL. As RCL contains the best individuals evaluated by fitness, these communities are considered promising regions.

The similarity level, r, between two individuals x and y is calculated by the Pearson correlation coefficient (Rodgers and Nicewander 1988) and is given by Eq. (20), where x and y are the random-keys vectors, and $\overline{x}$ and $\overline{y}$ are the arithmetic means of x and y, respectively. A adaptive parameter σ is defined by Eq. (21) and used to represent the similarity degree during the search process. If $r \geq \sigma$ then the individuals x e y are similar. The parameter σ starts in 0.3 and increases during the population evolution, because the diversity decreases throughout the evolution process.

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}. \tag{20}$$

$$\sigma = 0.30 + \frac{g_k}{max_{gen}} * (0.70 - 0.30). \tag{21}$$

Initially, we build a graph $G(V, A)$, where each individual of the RCL is represented by a node $v \in V$. There is an edge $a_{uv} \in A$ only if the individuals u and v have a similarity level greater than σ. Then to each node we assign a different number, named *label*. For each node v of the graph, LP searches the *label* with the highest frequency among its neighbors. If ties occur one of them is chosen at random. The label of v is then replaced by the label of this neighbor with the highest frequency. The algorithm ends when the *label* of each node in the graph is the more common of its neighborhood. Finally, all nodes with the same *label* are grouped into a community (cluster). Figure 3 illustrates an example of the LP applied to a RCL with 9 individuals. Every edge in this graph is incident to nodes having similar individuals. Starting with graph of Fig. 3a, LP ends up with the graph of Fig. 3d. The way the *labels* of the nodes are updated is shown one at a time. At the end, two clusters with *labels* 2 and 6 are obtained.

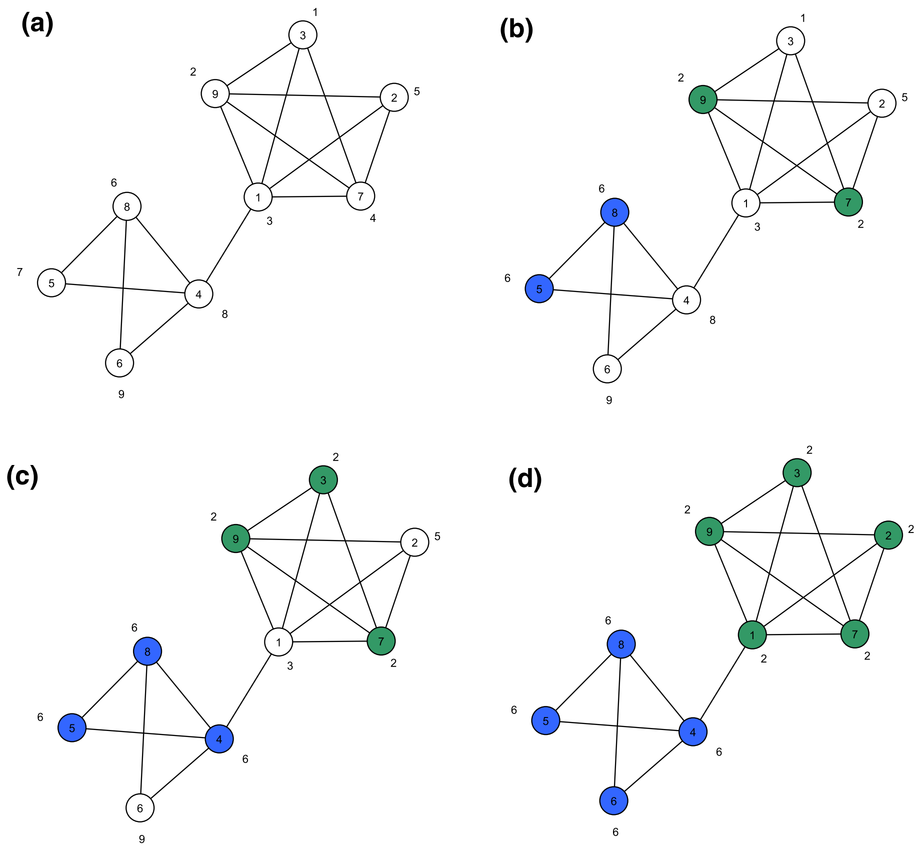

Fig. 3 Example of an iteration of the Label Propagation algorithm

The individual in a cluster with the best fitness value and that was not yet explored by the specific local search is chosen as the representative solution of this cluster. This individual is decoded into a solution of the problem and the local search is applied to this solution. The local optimal solution found does not return to the current population since this would require a re-decoding process. If this solution is better than the best-known solution so far, then the current best solution is updated. Since this solution is not returned to the current population, it does not interfere in the evolutionary process of the BRKGA or the A-BRKGA.

The BRKGA and A-BRKGA methods with this local search heuristics are named BRKGA+CS and A-BRKGA+CS, respectively (Chaves et al. 2016, 2018).

4 BRKGA+CS and A-BRKGA+CS for the MTSPPP

In this paper, we encode the solution of the MTSPPP as a vector S with n random-keys in the BRKGA+CS, and a vector of $n + 2$ random-keys in the A-BRKGA+CS, where n is the number of customers to be visited plus the depot, and the positions $n + 1$ and $n + 2$ are used to calculate the value of the parameters ρ_e and β, respectively. The decoding of S in a solution S' of MTSPPP is accomplished by sorting the customers in ascending order of their

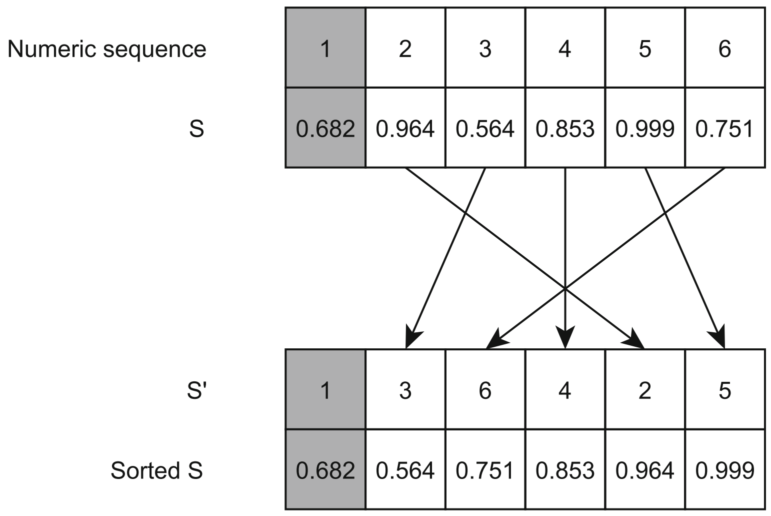

Fig. 4 Example of an encoded solution

corresponding random-key values. The first position in S is set at 1, representing the depot, whose salesman starts and ends his trip there. In Fig. 4, we show an example of the decoding process for the MTSPPP with six customers, where the gray rectangle represents the depot with its respective random-key value.

The fitness of Solution S' is calculated by the corresponding value of the objective function (1).

4.1 Local search for MTSPPP

The local search heuristic used is an Iterated Local Search (ILS) algorithm (Lourenço et al. 2003). ILS combines a local search phase with a perturbation phase. Specifically, the Variable Neighborhood Descent (VND) (Hansen et al. 2010) was used in the local search phase of ILS and Swap(1,1) was used in the perturbation phase.

Our VND used three neighborhood structures:

1. $N^{(1)}$—2-Opt: reverse the order of the visitation of customers between positions i and j, inclusive, in Solution S';
2. $N^{(2)}$—Shift(1): transfer customer i from its current position to position k, in Solution S';
3. $N^{(3)}$—Swap(1,1): change the visitation position of customer i with the visitation position of customer j in Solution S'.

Examples of these neighborhoods are shown in Fig. 5. In (a) we present an example of a route with 5 customers plus the depot whose order is 1–2–3–4–5–6. In (b) we present the route 1–5–4–3–2–6 obtained by applying 2-Opt to the positions 2 and 5. In (c) we present the route 1–3–4–5–2–6 obtained by applying Shift(1) to customer 2 to position 5. In (d) we present the route 1–5–3–4–2–6 obtained by applying Swap(1,1) to customers 2 and 5.

Let S' be a decoded representative solution, i.e., a solution provided by the clustering process and decoded by the decoder function. For each of these solutions, we apply Algorithms 1 and 2, which are the ILS and VND procedures, respectively.

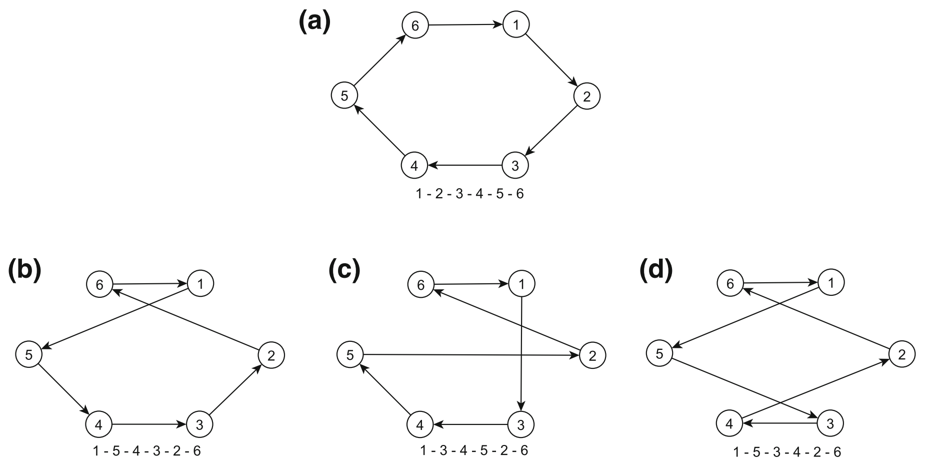

Fig. 5 Examples of the three neighborhood structures used in the MTSPPP. In **a** an example of a customer route. In **b** route obtained by 2-Opt. In **c** route obtained by Shift(1). In **d** route obtained by Swap(1,1)

Algorithm 1 Pseudo code of ILS

```
1: Initial solution $s_0$                          ▷ representative decoded solution of region
2: $s \leftarrow$ VND($s_0$)
3: $Iter$                                          ▷ current iteration
4: $Iter_{max} \leftarrow 3$                       ▷ maximum number of iterations
5: while ($Iter < Iter_{max}$) do
6:     Iter ← Iter + 1
7:     $s' \leftarrow$ Swap(1, 1)
8:     $s'' \leftarrow$ VND($s'$)
9:     if ($f(s'') > f(s)$) then
10:        $s \leftarrow s''$
11: Return $s$
```

Algorithm 2 Pseudo code of VND

```
1: Initial solution $s_0$
2: Number of neighborhood structures $n_r$
3: $s \leftarrow s_0$                              ▷ current iteration
4: $k \leftarrow 1$                                ▷ type of neighborhood structure
5: while ($k \leq n_r$) do
6:     Find the best neighbor $s' \in N^{(k)}$.
7:     if ($f(s') > f(s)$) then
8:         $s \leftarrow s'$
9:         $k \leftarrow 1$
10:    else
11:        $k \leftarrow k + 1$
12: Return $s$
```

In Algorithm 1, the initial solution (s_0) is a decoded solution (S') in a promising region determined as described in subsection 3.2. VND is applied to obtain a better solution that will be the current solution, s, (line 2). We set to 3 the number of iterations of the algorithms (line 4) because of the available running time. In lines 5–10, we apply the local search phase; in line 7 we realize a perturbation in the current solution by Swap(1,1) before applying VND. If a better solution is found then the current solution is updated (lines 9 and 10). This is recursively applied until the maximum number of iterations is reached.

In Algorithm 2, the initial solution s_0 is obtained by the ILS procedure (Algorithm 1, lines 2 and 8). The number of neighborhood structures n_r was set to 3, which are 2-opt, Shift(1) and Swap(1,1). A search is performed in all neighborhoods until the best solution for all structures is obtained (lines 5–11). In each neighborhood structure k (line 5) we find the best neighbor (line 6). Whenever a better solution is found, we return to the first neighborhood structure (line 9), ensuring that the final solution obtained is the best for all neighborhood structures. When a better solution is not found using the kth neighborhood structure (lines 10 and 11) then the structure changes to $k + 1$.

5 Computational results

The BRKGA+CS and A-BRKGA+CS were coded in C++ and the computational tests were carried out on an Intel Core i7 3.6 GHz processor with 16 GB of RAM in a Windows 10 environment. The solver ILOG CPLEX 12.6.3.0 (ILOG 2006) was used to obtain a solution using the model MTSPPP. The two versions of BRKGA were run using 20 different seeds.

The parameter tuning of BRKGA+CS is realized by the Relative Deviation Index (RDI) (Kim 1993). The RDI_i of a solution i with objective function value S_i is defined as

$$\text{RDI}_i = \frac{S_\text{B} - S_i}{S_\text{B} - S_\text{W}}, \tag{22}$$

where S_B is the best value obtained by the metaheuristics and S_W is the worst value obtained by the metaheuristics. Different combinations of parameters were tested on a subset of instances and the configuration with the best RDI was chosen for the computational tests of the BRKGA+CS.

The parameter's sets used in the tuning phase are shown in Table 1. The parameters found are given in Table 2 together with the range of the parameters used in A-BRKGA+CS. The settings of BRKGA+CS were kept constant for all instances whereas the parameters of A-BRKGA+CS varied within those intervals according to the theory presented in Sect. 3.1.

Table 1 Parameter values used in tuning process of BRKGA+CS

Parameter	Meaning	Values
p	Size of population	1000
p_e	Size of elite population	0.10, 0.15, 0.20, 0.25, 0.30
p_m	Size of mutant population	0.15, 0.20, 0.25, 0.30
ρ_e	Elite allele inheritance probability	0.70, 0.75, 0.80, 0.85

Table 2 Parameters used in the BRKGA+CS and the A-BRKGA+CS

Parameter	BRKGA+CS	A-BRKGA+CS
	Value	Range values
p	1000	[100, 1000]
Gen	300	271
p_e	0.15	[0.10, 0.25]
p_m	0.20	[0.05, 0.20]
ρ_e	0.70	[0.65, 0.80]

Table 3 The symmetric instances selected from the TSPLIB (TSPLib 2019), by instance sizes

1–50 customers	51–100 customers	101–150 customers
burma14	eil51	eil101
ulysses16	berlin52	lin105
gr17	brazil58	pr107
gr21	st70	gr120
ulysses22	eil76	pr124
gr24	pr76	bier127
fri26	gr96	ch130
bays29	rat99	
dj38	kroA100	
dantzig42	kroB100	
att48		
gr48		
hk48		

The instance dj38 was selected from TSP Test Data (2019)

The instances used in the tests were generated from the TSPLib library (2019) and were based on Sarubbi (2008). Each customer had an integer demand that ranged between 1 and τ (maximum demand), in which τ varied between 5 and 20 units. The depot, or initial node, always had a demand of 1 unit. The demands could be any positive real number. In this case, the value of the variables f_{ijl} in an optimal solution will be a real number. The product $\gamma_l * c_{ij}$ represents the cost of passing the product l through arc (i, j). The better the conditions for a arc (i, j) are, the lower the value of the corresponding parameter. This parameter γ_l varied from 0.5 to 2%. Each customer also had a η_l value correspondent. This corresponds to the relative value of the product, and the more valuable the product is, the greater the value of the parameter. Parameters η_l were randomly generated between 0.5 and 1.5. The priority prizes were randomly generated between 0 and 100, considering only integer values (Silva 2017). All these parameters were generated using a uniform distribution. In A-BRKGA, we set γ at 0.999 because the running time implied by it was enough for the metaheuristic to find satisfactory solutions.

All instances satisfied the triangular inequality in relation to fixed costs. Each instance has its number of customers specified in its name. For example, instance burma14 has 13 customers to visit and the depot. In Table 3, all instances used in the computational tests are presented divided by the quantity of customers.

We generated three sets of instances that differ themselves by a θ parameter multiplied to the priority prizes. This parameter serves only to generate different instances. We denoted the set with $\theta = \overline{c}$ by A, the set with $\theta = \frac{\overline{c}}{4(N-1)\overline{p}}\left(2N + \sum_{l=2}^{N} z_l\right)$ by B, and the set with $\theta = 1$ by C, where $\overline{c} = \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N} c_{ij}$, $\overline{p} = \frac{1}{(N-1)^2}\sum_{k=2}^{N}\sum_{i=2}^{N} p_{ki}$, and $z_l = \gamma_l \eta_l q_l$.

Set A is closest to the assignment problem, since the priority prizes are much larger than the fixed costs. Set B is a balance between the assignment problem and the traveling salesman problem, and Set C is closest to the traveling salesman problem since the priority prizes are much smaller than the fixed costs.

In Tables 4, 5 and 6, we present the results for the CPLEX, A-BRKGA+CS and BRKGA+CS for Sets A, B, and C, respectively. The entries in the tables are the instances; the solution obtained by the CPLEX (sol$_C$) within time limit of 10,800 s, the best solution

(S^*), the average solution (S) over 20 runs, the time to find the best solution (T^*), the average running time (T) in seconds, the deviation (Dev) defined as $100 \times (S - S^*)/S^*$, and the GAP provided by CPLEX defined as ($|upperbound - lowerbound|/|lowerbound|$). The entries "-" in the column sol$_C$ mean that the CPLEX was not able to find a feasible solution within the time limit set.

In Table 7, we present the results for the linear relaxation compared to the solutions of the proposed metaheuristics. The gaps were calculated using Eq. (23), where S_{RL} is the solution of the linear relaxation, S_{C} is the solution of the integer model from CPLEX, S_{A} is the solution from A-BRKGA+CS, and S_{B} is the solution from BRKGA+CS. In Time(s) column, the times used by CPLEX to solve the linear relaxation in instances of Sets A, B, and C are presented.

$$\mathrm{GAP}^{i}_{\mathrm{RL}} = \frac{|S_{\mathrm{RL}} - S_i|}{|S_{\mathrm{RL}}|}, \ i = \mathrm{A, B, C}. \tag{23}$$

CPLEX was not able to find a feasible solution to all instances within the time limit set. For set A, CPLEX was able to solve 8 instances to optimality within the limit time set, and 6 instances with GAPs smaller than 4%. But, it was unable to find a feasible solution in 14 of the 30 instances. For Sets B and C, CPLEX solved only 2 instances to optimality and was not able to find feasible solutions in 13 instances.

When the priority prizes were much larger than the costs (Set A), the assignment problem gains dominated the overall objective function and the model was able to solve more instances to optimality. The assignment problem has the integrality property, i.e., the solution of linear relaxation is the optimal solution of the problem, which may explain the smaller execution time for these instances compared to the instances in Sets B and C. As a consequence, the linear relaxation solution MTSPPP model approached the optimal solution. In Table 7, the values of $\mathrm{GAP}^{\mathrm{C}}_{\mathrm{RL}}$ show the relative distance between the linear relaxation and the optimal solution (or better solution provided by CPLEX). Six instances had $\mathrm{GAP}^{\mathrm{C}}_{\mathrm{RL}}$ smaller than 4.5%, not counting instances `eil51` and `berlin52`, where the solutions found within the time limit by CPLEX were not very good, or others where the CPLEX did not find a feasible solution,

In Sets B and C, the linear relaxation solution did not present good results, producing gaps around 100%, and, in some cases, even larger gaps, as in the cases of instances `gr48`, `berlin52`, and `brazil58`, with gaps of 3742.34%, 1883.20%, and 1157.06%, respectively, in Set C.

In Tables 4, 5, and 6, we observe that the A-BRKGA+CS performed worse than the BRKGA+CS for the instances of Set A. The first metaheuristic was better in 30% of the instances, while the second one was better in 43%. Both metaheuristics found the same solutions in 26% of the instances. In Set B, the A-BRKGA+CS was better in 46% of the instances, while the BRKGA+CS was better in 23%. In Set C, the A-BRKGA+CS was better in 33% of the instances, while BRKGA+CS was better in 20%. Observe that, when priority prize penalties decrease, the number of instances in which the metaheuristics return the same solution increases from 26% in A to 30% and 46% in B and C.

To compare the two sets of solutions given by A-BRKGA+CS and BRKGA +CS for each set and analyze if there is a significant difference between them, the Wilcoxon signed-rank (WSR) test (Wilcoxon 1945) and Friedman test (Friedman 1940) were applied. The results of the tests are shown in Table 8 where Z is the sum of the flagged posts of the WSR test, Chi squared is the test statistic and df is the degrees of freedom of the Friedman test. We concluded that the A-BRKGA+CS and BRKGA+CS ranks were statistically equivalent (WSR test) and the distributions of the scores for the methods compared are equal (p value > 0.001—Friedman test) for all sets, at a 0.05 significance level.

Table 4 Results from CPLEX, BRKGA+CS, and A-BRKGA+CS for Set A

Instance	CPLEX				A-BRKGA+CS					BRKGA+CS				
	sol$_C$	T^*	T	GAP (%)	S^*	S	T^*	T	Dev	S^*	S	T^*	T	Dev
burma14	**540,978.745**	0.16	0.17	0.00	**540,978.745**	540,978.75	0.06	5.97	0.00	**540,978.745**	540,978.75	0.08	5.32	0.00
ulysses16	**1,045,426.76**	0.52	0.55	0.00	**1,045,426.76**	1,045,426.56	0.07	7.96	0.00	**1,045,426.76**	1,045,426.56	0.08	6.07	0.00
gr17	**391,773.68**	0.44	0.47	0.00	**391,773.68**	391,773.66	0.06	8.94	0.00	**391,773.68**	391,773.66	0.13	6.87	0.00
gr21	**667,131.56**	32.67	38.64	0.00	**667,131.56**	667,131.56	5.90	16.18	0.00	**667,131.56**	667,032.30	5.83	10.95	– 0.01
ulysses22	**1,408,527.99**	75.59	92.73	0.00	**1,408,527.99**	1,408,433.00	9.21	16.75	– 0.01	**1,408,527.99**	1,407,771.00	0.67	12.50	– 0.05
gr24	**313,651.42**	5.89	48.70	0.00	**313,651.42**	313,619.10	9.33	22.18	– 0.01	**313,651.42**	313,535.60	16.53	15.48	– 0.04
fri26	**237,759.19**	206.00	223.45	0.00	**237,759.19**	237,690.10	1.54	28.41	– 0.03	**237,759.19**	237,624.30	12.32	18.41	– 0.06
bays29	**539,347.37**	36.31	378.92	0.00	**539,347.37**	538,961.70	14.83	38.43	– 0.07	**539,347.37**	538,981.00	13.70	28.95	– 0.07
dj38	2,468,189.09	9697.27	10,800.00	1.85	2,471,873.25	2,466,715.00	56.85	87.48	– 0.21	**2,472,919.00**	2,466,184.00	68.47	85.12	– 0.27
dantzig42	**285,605.55**	1991.34	10,800.00	0.52	284,900.28	284,180.30	11.48	131.21	– 0.25	284,895.28	284,191.00	42.76	124.45	– 0.25
att48	4,559,007.06	9603.00	10,800.00	2.51	4,587,733.50	4,581,998.00	64.47	206.49	– 0.13	**4,597,805.00**	4,584,123.00	60.46	219.78	– 0.30
gr48	**1,935,290.36**	2480.66	10,800.00	1.22	1,929,587.50	1,921,895.00	86.37	209.56	– 0.40	1,926,223.75	1,921,710.00	10.55	232.50	– 0.23
hk48	**4,511,601.81**	7251.27	10,800.00	1.32	4,501,591.50	4,488,540.00	89.82	182.92	– 0.29	4,501,749.00	4,489,545.00	114.54	197.17	– 0.27
eil51	85,482.75	9967.59	10,800.00	44.99	152,334.92	151,913.60	147.67	250.58	– 0.28	**152,357.78**	151,925.60	209.69	261.20	– 0.28
berlin52	1,235,108.12	4929.02	10,800.00	55.86	2,722,884.50	2,718,976.00	13.28	267.92	– 0.14	**2,729,865.75**	2,720,083.00	178.67	323.06	– 0.36
brazil58	11,281,984.18	9634.48	10,800.00	3.87	11,461,121.00	2,718,976.00	117.48	385.85	– 76.28	**11,463,258.00**	11,432,682.00	35.14	462.65	– 0.27

Table 4 continued

Instance	CPLEX				A-BRKGA+CS					BRKGA+CS				
	sol$_C$	T*	T	GAP (%)	S*	S	T*	T	Dev	S*	S	T*	T	Dev
st70	–	–	10,800.00	–	**338,038.62**	337,051.00	1091.21	854.01	– 0.29	337,896.59	337,118.70	274.37	880.66	– 0.23
eil76	–	–	10,800.00	–	236,625.89	236,198.70	721.98	914.20	– 0.18	**236,852.78**	236,179.50	276.21	923.28	– 0.28
pr76	–	–	10,800.00	–	52,913,480.00	52,733,805.00	1414.22	1163.21	– 0.34	**52,959,868.00**	52,768,027.00	604.74	1388.64	– 0.36
gr96	–	–	10,800.00	–	33,820,656.00	33,737,586.00	487.36	1500.00	– 0.25	**33,830,080.00**	33,737,084.00	293.76	1500.00	– 0.27
rat99	–	–	10,800.00	–	**782,456.37**	780,396.80	50.38	1500.00	– 0.26	781,264.31	780,111.00	1384.77	1500.00	– 0.15
kroA100	–	–	10,800.00	–	**16,027,485.00**	15,995,068.00	1054.29	1494.67	– 0.20	16,007,513.00	15,993,466.00	801.31	1500.00	– 0.09
kroB100	–	–	10,800.00	–	15,507,184.00	15,458,129.00	696.23	1500.00	– 0.32	**15,517,892.00**	15,464,435.00	636.95	1500.00	– 0.34
eil101	–	–	10,800.00	–	**317,354.22**	316,515.60	10.79	1500.00	– 0.26	316,745.34	316,341.80	1020.83	1500.00	– 0.13
lin105	–	–	10,800.00	–	11,429,048.00	11,413,877.00	583.18	1500.00	– 0.13	**11,441,579.00**	11,419,229.00	1325.33	1500.00	– 0.20
pr107	–	–	10,800.00	–	**54,178,324.00**	54,063,147.00	448.93	1500.00	– 0.21	54,177,276.00	54,071,862.00	1500.00	1500.00	– 0.19
gr120	–	–	10,800.00	–	4,892,669.50	4,885,372.00	395.97	1500.00	– 0.15	**4,896,112.50**	4,886,659.00	1185.42	1500.00	– 0.19
pr124	–	–	10,800.00	–	**65,399,888.00**	65,259,224.00	1021.24	1500.00	– 0.22	65,371,232.00	65,263,581.00	574.86	1500.00	– 0.16
bier127	–	–	10,800.00	–	**57,905,416.00**	57,718,236.00	1364.71	1500.00	– 0.32	57,894,204.00	57,706,235.00	250.94	1500.00	– 0.32
ch130		–	10,800.00	–	4,246,087.50	4,234,849.00	1121.07	1500.00	– 0.26	**4,254,836.00**	4,237,412.00	841.52	1500.00	– 0.41

Table 5 Results from CPLEX, BRKGA+CS, and A-BRKGA+CS for Set B

Instance	CPLEX				A-BRKGA+CS					BRKGA+CS				
	sol$_C$	T^*	T	GAP (%)	S^*	S	T^*	T	Dev	S^*	S	T^*	T	Dev
`burma14`	− **450.99**	7.84	23.23	0.00	− **450.99**	− 450.99	0.05	5.95	0.00	− **450.99**	− 450.99	0.07	5.06	0.00
`ulysses16`	**398.77**	1471.41	1645.98	0.00	**398.77**	398.77	0.10	8.21	0.00	**398.77**	398.77	0.20	5.88	0.00
`gr17`	− **255.93**	5730.20	10,800.00	222.31	− **255.93**	− 255.93	0.55	8.52	0.00	− **255.93**	− 255.93	0.10	6.22	0.00
`gr21`	352.03	10,660.22	10,800.00	634.40	**1036.21**	1036.21	0.36	14.69	0.00	**1036.21**	1036.21	0.10	8.94	0.00
`ulysses22`	− 1815.79	10,763.17	10,800.00	403.05	− **114.06**	− 114.06	0.21	16.90	0.00	− **114.06**	− 114.06	0.50	10.23	0.00
`gr24`	422.16	10258.19	10,800.00	240.77	**592.70**	592.70	0.11	20.17	0.00	**592.70**	592.70	1.05	11.83	0.00
`fri26`	− 584.55	504.84	10,800.00	254.84	**154.16**	154.16	3.62	25.24	0.00	**154.16**	152.16	7.90	13.94	− 1.30
`bays29`	− 792.27	994.81	10,800.00	465.78	**1033.42**	1033.42	10.57	36.20	0.00	**1033.42**	1033.42	2.30	18.51	0.00
`dj38`	− 14,921.94	4261.42	10,800.00	200.18	**79.70**	48.17	63.02	78.82	− 39.55	**79.70**	59.99	27.56	39.37	− 24.73
`dantzig42`	− 377.75	2747.42	10,800.00	665.69	**1181.95**	1170.08	5.52	119.11	− 1.00	**1181.95**	1161.66	58.72	67.77	− 1.72
`att48`	− 34,997.32	5680.05	10,800.00	202.74	**11,509.23**	11,000.21	26.74	180.89	− 4.42	11,384.19	10,618.18	4.15	93.24	− 6.73
`gr48`	− 15,186.18	6600.11	10,800.00	195.60	**3222.88**	2968.09	142.21	170.24	− 7.91	3120.87	2846.64	41.52	89.26	− 8.79
`hk48`	− 28,065.98	9041.84	10,800.00	217.55	**10,385.87**	9691.58	73.52	188.80	− 6.68	10,132.88	9507.07	83.61	96.82	− 6.18
`eil51`	− 2021.21	9783.72	10,800.00	151.81	− **4.75**	− 22.89	55.46	208.29	381.89	− 6.71	−31.93	107.82	117.95	375.86
`berlin52`	− 48,972.09	7874.70	10,800.00	140.29	**150.08**	− 228.77	75.36	209.51	− 252.43	104.42	− 225.55	88.93	109.00	− 316.00
`brazil58`	− 189,031.01	3904.91	10,800.00	150.20	**24,532.86**	22,405.51	172.21	333.56	− 8.67	23,626.16	21,742.41	129.32	171.02	− 7.97

Table 5 continued

Instance	CPLEX				A-BRKGA+CS					BRKGA+CS				
	sol$_C$	T*	T	GAP (%)	S*	S	T*	T	Dev	S*	S	T*	T	Dev
`st70`	− 9299.25	10,800.00	10,800.00	130.54	− 76.13	−124.23	128.42	669.21	63.18	− **49.33**	− 125.55	159.07	365.88	154.51
`eil76`	–	–	10,800.00	–	672.78	645.64	135.41	849.89	− 4.03	**676.98**	633.26	342.44	570.33	− 6.46
`pr76`	–	–	10,800.00	–	− **128,443.44**	− 157,740.40	329.10	730.39	22.81	− 138,540.61	− 165,986.80	278.70	444.19	19.81
`gr96`	–	–	10,800.00	–	− **3591.47**	− 17,948.34	878.12	1344.83	399.75	− 10,091.05	− 18,797.72	241.18	1176.87	86.28
`rat99`	–	–	10,800.00	–	− **39.85**	− 215.76	675.15	1454.31	441.43	− 84.85	−247.05	270.28	1428.41	191.16
`kroA100`	–	–	10,800.00	–	**45,349.34**	42,688.74	773.23	1403.66	− 5.87	45,022.53	41,844.28	81.78	1295.84	− 7.06
`kroB100`	–	–	10,800.00	–	− **12,458.55**	− 15,460.95	582.39	1392.15	24.10	− 12,964.93	− 15,100.84	228.66	1259.72	16.47
`eil101`	–	–	10,800.00	–	− 324.46	−398.00	707.71	1481.23	22.67	− **297.43**	− 393.02	1268.13	1456.59	32.14
`lin105`	–	–	10,800.00	–	12,579.35	10,738.05	210.91	1500.00	− 14.64	**14557.37**	11767.65	298.28	1450.78	− 19.16
`pr107`	–	–	10,800.00	–	258,159.51	247,378.60	1320.24	1428.66	− 4.18	**263,117.72**	249,222.40	1445.38	1471.89	− 5.28
`gr120`	–	–	10,800.00	–	**7569.26**	6243.56	282.75	1500.00	− 17.51	7110.58	6047.70	118.76	1500.00	− 14.95
`pr124`	–	–	10,800.00	–	251,567.62	237,369.20	1167.08	1500.00	− 5.64	**259,163.23**	235,636.60	623.12	1396.29	− 9.08
`bier127`	–	–	10,800.00	–	− **190,174.28**	− 199,969.60	980.34	1500.00	5.15	− 193,390.01	− 204,634.10	616.58	1500.00	5.81
`ch130`	–	–	10,800.00	–	− **13,712.49**	− 14,965.97	772.34	1500.00	9.14	− 14,051.79	− 15,269.40	601.10	1500.00	8.67

Table 6 Results from CPLEX, BRKGA+CS, and A-BRKGA+CS for Set C

Instance	CPLEX				A-BRKGA+CS					BRKGA+CS				
	sol$_C$	T^*	T	GAP (%)	S^*	S	T^*	T	Dev	S^*	S	T^*	T	Dev
burma14	− **4382.58**	35.95	37.80	0.00	− **4382.58**	− 4382.58	0.05	5.59	0.00	− **4382.58**	− 4382.58	0.07	4.89	0.00
ulysses16	− **7552.15**	38.16	2027.34	0.00	− **7552.15**	− 7552.15	0.07	7.05	0.00	− **7552.15**	− 7552.15	0.09	5.63	0.00
gr17	− 2533.05	4154.95	10,800.00	17.57	− **2518.18**	− 2518.18	0.06	8.74	0.00	− **2518.18**	− 2518.18	0.49	6.06	0.00
gr21	− 2952.52	9950.06	10,800.00	43.08	− **2793.71**	− 2793.71	0.08	12.42	0.00	− **2793.71**	− 2793.71	0.11	8.49	0.00
ulysses22	− 11,616.90	10,765.53	10,800.00	32.64	− **10,726.12**	− 10,726.12	0.09	17.15	0.00	− **10,726.12**	− 10,726.12	0.25	9.63	0.00
gr24	− 709.25	10,784.16	10,800.00	152.20	− **381.60**	− 381.60	1.15	18.30	0.00	− **381.60**	− 381.80	0.18	11.57	0.05
fri26	− 544.74	10,800.00	10,800.00	221.74	− **54.05**	− 54.05	1.30	25.14	0.00	− **54.05**	− 54.05	7.34	13.81	0.00
bays29	− 2454.39	10,358.38	10,800.00	94.92	− **1526.08**	− 1526.08	0.67	30.63	0.00	− **1526.08**	− 1526.08	10.56	17.09	0.00
dj38	− 33,628.32	7093.38	10,800.00	77.21	− **16,323.99**	− 16,323.99	0.44	67.68	0.00	− **16,323.99**	− 16,323.99	0.23	33.24	0.00
dantzig42	133.32	3219.05	10,800.00	2129.47	**1944.54**	1923.37	126.69	121.21	− 1.09	**1944.54**	1913.10	41.58	73.45	− 1.62
att48	− 51,523.41	8275.50	10,800.00	86.15	− **23,027.95**	− 23,106.67	3.98	155.08	0.34	− **23,027.95**	− 23,138.10	12.12	77.35	0.48
gr48	− 25,912.34	8749.89	10,800.00	95.59	− **9327.03**	− 9345.08	1.07	141.69	0.19	− **9327.03**	− 9346.21	9.10	75.75	0.21
hk48	− 48,439.47	9783.30	10,800.00	82.37	− **22,170.77**	− 22,198.52	49.06	139.38	0.13	− **22,170.77**	− 22,243.41	57.04	71.20	0.33
eil51	403.81	10,431.31	10,800.00	919.73	**2699.44**	2646.44	174.89	258.03	− 1.96	2686.90	2617.99	86.03	129.27	− 2.56
berlin52	− 64,634.27	9877.58	10,800.00	93.95	− **18,296.24**	− 18,352.20	167.28	221.52	0.31	− 18,317.84	− 18,456.44	37.59	86.54	0.76
brazil58	− 209,329.54	6962.56	10,800.00	91.30	− **61,384.78**	− 61,661.89	216.27	303.81	0.45	− 61,384.78	− 61,688.62	41.83	158.59	0.49

Table 6 continued

Instance	CPLEX				A-BRKGA+CS					BRKGA+CS				
	sol$_C$	T*	T	GAP (%)	S*	S	T*	T	Dev	S*	S	T*	T	Dev
`st70`	− 7354.87	10,800.00	10,800.00	178.10	2370.46	2191.65	177.29	611.87	− 7.54	**2390.72**	2162.09	181.43	373.85	− 9.56
`eil76`	–	–	10,800.00	–	**4914.68**	4821.67	720.53	734.54	− 1.89	4898.41	4814.25	792.84	720.51	− 1.72
`pr76`	–	–	10,800.00	–	− **524,359.37**	− 530,491.80	343.97	689.38	1.17	− 526,192.50	− 531,420.90	110.46	396.56	0.99
`gr96`	–	–	10,800.00	–	− 244,170.83	− 248,055.80	1087.66	1308.98	1.59	− **243,754.44**	− 248,103.70	26.86	1118.25	1.78
`rat99`	–	–	10,800.00	–	**735.25**	522.96	505.07	1447.80	− 28.87	632.99	518.22	48.93	1404.39	− 18.13
`kroA100`	–	–	10,800.00	–	− **64,474.61**	− 65,007.84	198.95	1257.57	0.83	− 64,532.23	− 65,066.75	274.64	1170.08	0.83
`kroB100`	–	–	10,800.00	–	− **117,031.14**	− 118,457.90	541.86	1287.98	1.22	− 117,720.92	− 118,632.80	786.13	1232.18	0.77
`eil101`	–	–	10,800.00	–	4656.28	4596.09	397.79	1483.00	− 1.29	**4745.54**	4600.92	1211.42	1319.26	− 3.05
`lin105`	–	–	10,800.00	–	− 62,598.05	− 63,369.07	1088.46	1333.40	1.23	− **61,896.99**	− 63,097.24	281.51	1427.69	1.94
`pr107`	–	–	10,800.00	–	− 163,506.89	− 165,193.50	818.66	1376.97	1.03	− **163,872.11**	− 165,041.10	31.42	1426.13	0.71
`gr120`	–	–	10,800.00	–	− **20,219.13**	− 21,148.39	580.17	1500.00	4.60	20,537.48	− 21,140.62	1393.47	1500.00	− 202.94
`pr124`	–	–	10,800.00	–	− 196,417.47	− 200,211.80	1474.34	1500.00	1.93	− **195,745.12**	− 200,318.80	73.23	1500.00	2.34
`bier127`	–	–	10,800.00	–	− 563,419.06	− 572,220.80	1070.57	1500.00	1.56	− **566,039.37**	− 572,754.50	1074.70	1500.00	1.19
`ch130`	–	–	10,800.00	–	− 35,434.74	− 36,368.38	244.92	1500.00	2.63	− **35,063.98**	− 36,300.84	1278.32	1500.00	3.53

Table 7 Results for the linear relaxation of Sets A, B, and C

Instance	Set A				Set B				Set C			
	GAP^{C}_{RL} (%)	GAP^{A}_{RL} (%)	GAP^{B}_{RL} (%)	Time (s)	GAP^{C}_{RL} (%)	GAP^{A}_{RL} (%)	GAP^{B}_{RL} (%)	Time (s)	GAP^{C}_{RL} (%)	GAP^{A}_{RL} (%)	GAP^{B}_{RL} (%)	Time (s)
burma14	0.22	0.22	0.22	0.12	119.77	119.77	119.77	1.64	72.76	72.76	72.76	0.04
ulysses16	0.01	0.01	0.01	0.35	92.01	92.01	92.01	0.05	53.35	53.35	53.35	0.06
gr17	0.19	0.19	0.19	0.72	112.06	112.06	112.06	0.06	200.05	198.29	198.29	0.07
gr21	0.55	0.55	0.55	1.17	90.90	73.21	73.21	0.15	188.83	173.30	173.30	0.13
ulysses22	1.07	1.07	1.07	1.59	119.99	101.26	101.26	0.14	135.90	117.81	117.81	0.14
gr24	0.36	0.36	0.36	2.08	78.27	69.49	69.49	0.24	186.79	146.70	146.70	0.75
fri26	0.59	0.59	0.59	3.74	147.00	87.60	87.60	2.11	154.80	105.44	105.44	2.15
bays29	0.54	0.54	0.54	5.90	122.37	70.82	70.82	0.58	609.51	416.80	416.80	0.55
dj38	2.31	2.16	2.12	35.85	179.79	99.57	99.57	1.39	564.35	222.49	222.49	1.53
dantzig42	0.81	1.05	1.05	45.53	116.00	49.93	49.93	1.13	95.90	40.13	40.13	1.11
att48	2.94	2.33	2.12	85.63	189.79	70.47	70.79	3.02	676.70	247.14	247.14	1.88
gr48	1.50	1.79	1.96	431.68	196.35	79.55	80.20	14.10	3742.34	1283.03	1283.03	5.39
hk48	1.59	1.81	1.81	273.59	179.40	70.62	71.33	4.38	506.37	177.54	177.54	4.03
eil51	45.15	2.26	2.24	400.22	277.89	100.42	100.59	3.31	90.64	37.42	37.71	2.55
berlin52	56.01	3.03	2.78	1105.91	330.44	99.29	99.51	6.42	1883.20	461.39	462.05	5.89
brazil58	4.14	2.61	2.60	572.05	289.09	75.46	76.37	16.05	1157.06	268.63	268.63	11.55

Table 7 continued

Instance	Set A				Set B				Set C			
	GAP^{C}_{RL} (%)	GAP^{A}_{RL} (%)	GAP^{B}_{RL} (%)	Time (s)	GAP^{C}_{RL} (%)	GAP^{A}_{RL} (%)	GAP^{B}_{RL} (%)	Time (s)	GAP^{C}_{RL} (%)	GAP^{A}_{RL} (%)	GAP^{B}_{RL} (%)	Time (s)
st70	–	3.30	3.34	1170.60	406.77	102.51	101.63	15.34	223.23	60.28	59.94	9.28
eil76	–	2.11	2.02	2102.69	–	64.98	64.77	19.21	–	26.94	27.19	26.20
pr76	–	4.02	3.94	3796.21	–	127.57	129.74	25.38	–	377.10	378.77	33.72
gr96	–	4.40	4.38	13,350.90	–	101.18	103.31	57.53	–	333.25	332.51	115.79
rat99	–	3.95	4.10	23,401.10	–	100.56	101.20	50.89	–	90.92	92.19	38.53
kroA100	–	3.42	3.54	13,896.50	–	69.99	70.20	49.99	–	435.84	436.32	96.37
kroB100	–	4.86	4.79	9882.29	–	108.34	108.68	121.51	–	807.95	813.30	140.18
eil101	–	4.13	4.31	13,409.00	–	111.70	110.73	78.29	–	48.52	47.54	87.04
lin105	–	4.62	4.51	16,668.70	–	89.14	87.43	101.11	–	6326.84	6254.86	127.93
pr107	–	3.69	3.69	12,816.00	–	53.18	52.28	59.36	–	734.29	736.15	93.32
gr120	–	3.89	3.82	100,319.00	–	83.21	84.22	215.40	–	565.48	– 372.81	198.73
pr124	–	3.62	3.66	118,643.00	–	61.55	60.39	135.76	–	484.36	482.36	124.63
bier127	–	5.45	5.47	79,501.30	–	136.23	136.84	448.98	–	425.33	427.77	566.28
ch130	–	5.93	5.74	111,147.00	–	132.72	133.53	236.33	–	666.67	660.74	348.63

Table 8 Result of the WSR test and Friedman test for Sets A, B, and C

WSR test			Friedman test		
Set	Z	*p* value	Chi-squared	*df*	*p* value
A	97	0.3464	0.7273	1	0.3938
B	164	0.0952	2.3333	1	0.1266
C	80	0.5521	1.0000	1	0.3173

From Tables 4, 5, and 6, we observe that the proposed MTSPPP model found better solutions than the proposed metaheuristics for three instances of set A: `dantzig42`, `gr48`, and `hk48`. The gaps of the solutions obtained by A-BRKGA+CS, for these instances were 0.76%, 1.51%, and 1.54%, respectively, and, for the solutions obtained by BRKGA+CS, they were 0.77%, 1.69%, and 1.54%, respectively. The solutions obtained by the models have gaps 0.52%, 1.22%, and 1.32%. On the other hand, the model found worse solutions for two instances, `eil51` and `berlin52`, showing gaps of 44.99% and 55.86%, respectively, while A-BRKGA+CS obtained solutions with gaps of 1.97% and 2.69%, and BRKGA+CS obtained solutions with gaps of 1.96% and 2.44%. For all the other instances in Sets B and C, the proposed model performed worse than the proposed heuristic methods.

6 Conclusion

In this paper, we presented a new variant of the Traveling Salesman Problem that considers variable costs and priority prizes, in addition to fixed costs. A mathematical model was proposed to represent this problem, using a multicommodity flow model. Two hybrid methods were proposed to find solutions to this problem. The first consisted of a Biased Random-Key Genetic Algorithm (BRKGA) with a Label Propagation (LP) community*** detection method. For the local search, an Iterated Local Search (ILS) combined with Variable Neighborhood Descent (VND) was proposed. The second method consisted of an Adaptive BRKGA with the same communities detection method and local search process.

For the computational experiments, 90 different instances were generated based on instances of the TSPLib library, TSP Test Data, and Sarubbi (2008). Three penalties to priority prizes were generated, resulting in three sets of instances: Set A consisted of penalizing the priority prizes for average fixed costs; Set B the penalty was using a balance between fixed costs and variable costs; Set C did not use any penalty.

The proposed model was able to provide good solutions for instances of Set A, but not for instances in Sets B and C. This happened because of the linear relaxation model. While in Set A the linear relaxation model was very good, approaching the optimal solution, in Sets B and C, it performed poorly. Because of penalties, the instances in Set A approached the assignment problem that has the integrality property, where the solution of linear relaxation is the optimum solution of the integer model.

Both the proposed heuristic methods performed well in both computational time and quality of solution of instances tested. In general, the A-BRKGA+CS method found better solutions (33) than the BRKGA+CS method (26), although it was not statistically different according to the Wilcoxon and Friedman tests performed. The main difference between the two methods is that the BRKGA+CS method needs to be calibrated, while the A-BRKGA+CS method is adaptive.

Future studies could be to develop other decoders in the proposed methods. For example, partial sequences of customers could be considered to build all the sequencing. Other local search methods also could be applied and methods to improve the upper bounds of the model could be developed. A re-decoding process can be studied to return the solution obtained by local search to the current population of BRKGA.

Acknowledgements The authors thank the anonymous reviewers for valuable comments and suggestions that improved the quality of the presentation of the paper. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES)—Finance Code 001, FAPESP (grant 2018/15417-8), and CNPq (grants 423694/2018-9, 303736/2018-6, and 303144/2014-9).

References

Angel RD, Caudle WL, Noonan R, Whinston A (1972) Computer assisted school bus scheduling. Manag Sci 18:279–88

Applegate DL, Bixby RE, Chvatal V, Cook WJ (2006) The traveling salesman problem: a computational study. Princeton University Press, Princeton

Archetti C, Speranza MG, Vigo D (2014) Vehicle routing problems with profits. In: Toth P, Vigo D (eds) Vehicle routing: problems, methods, and applications, 2nd edn, MOS-SIAM Series on optimization

Balas E (1987) The prize collecting travelling salesman problem. Networks 19:621–636

Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. ORSA J Comput 6:154–160

Chaves AA, Lorena LAN, Senne ELF, Resende MGC (2016) Hybrid method with CS and BRKGA applied to the minimization of tool switches problem. Comput Oper Res 67:174–183

Chaves AA, Gonçalves JF, Lorena LAN (2018) Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. Comput Ind Eng 124:331–346

Dantzig R, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling salesman problem. Oper Res 2:393–410

Flood MM (1956) The traveling-salesman problem. Oper Res 4(1):61–75

Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11(1):86–92

Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13(5):533–549

Gonçalves J, Resende M (2011) Biased random-key genetic algorithms for combinatorial optimization. J Heuristics 17:487–525

Gonçalves JF, Resende MGC (2016) Random-key genetic algorithms. Springer International Publishing, Handbook of Heuristics, Ch. 1, pp 1–13

Grotschel M, Junger M, Reinelt G (1991) Optimal control of plotting and drilling machines: a case study. Math Methods Oper Res 35(1):61–84

Hansen P, Mladenović N, Moreno Pérez JA (2010) Variable neighbourhood search: methods and applications. Ann Oper Res 175:367–407

ILOG Using the CPLEX Callable Library, Copyright, ILOG (2006)

Ingber L (1993) Simulated annealing: practice versus theory. Math Comput Model 18(11):29–57

Kim Y-D (1993) Heuristics for flowshop scheduling problems minimizing mean tardiness. J Oper Res Soc 44(1):19–28

Koopmans T, Beckmann MJ (1957) Assignment problems and the location of economic activities. Econometrica 25(1):53–76

Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1985) The traveling salesman problem: a guided tour of combinatorial optimization. Wiley, New York

Lourenço HR, Martin O, Stützle T (2003) Iterated local search. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer Academic Publishers, International Series in Operations Research & Management Science, pp 321–353

Lozano L, Smith JC, Kurz ME (2017) Solving the traveling salesman problem with interdiction and fortification. Oper Res Lett 45(3):210–216

Malaguti E, Martello S, Santini A (2018) The traveling salesman problem with pickups, deliveries, and draft limits. Omega 74:50–58

Prasetyo H, Fauza G, Amer Y, Lee SH (2015) Survey on applications of biased-random key genetic algorithms for solving optimization problems. In: International conference on industrial engineering and engineering management (IEEM), pp 863–870

Pureza V, Morabito R, Luna HP (2018) Modeling and solving the traveling salesman problem with priority prizes. Pesqui Oper 38(3):499–522
Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E APS 76(3):036106-1–036106-11
Ratliff HD, Rosenthal AS (1983) Order-picking in a rectangular warehouse: a solvable case for the traveling salesman problem. Oper Res 31(3):507–521
Rodgers JL, Nicewander WA (1988) Thirteen ways to look at the correlation coefficient. Am Stat Taylor Francis 42(1):59–66
Sarubbi JFM (2003) Um modelo linear para o problema do caixeiro viajante com demandas heterogêneas. Dissertation, Federal University of Minas Gerais
Sarubbi JFM (2008) Problemas de Roteamento com Custos de Carga. Ph.D Thesis, Federal University of Minas Gerais
Sarubbi JFM, Luna HPL (2007) The multicommodity traveling salesman problem. In: INOC–internacional network optimization conference, Belgian
Sarubbi JFM, Mateus GR, Luna HP, Miranda GD (2007) Model and algorithms for the multicommodity traveling salesman problem. In: 7th international conference on hybrid intelligent systems, Germany, pp 113–119
Sarubbi JFM, Miranda G, Luna HPL, Mateus G (2008) A cut-and-branch algorithm for the multicommodity traveling salesman problem. In: 2008 IEEE international conference on service operations and logistics and informatics, Beijing. Service operations and logistics and informatics, vol 2, pp 1806–1811
Silva AA (2017) Abordagens de Otimização para apoiar a Elaboração e Análise de Roteiros Turísticos. Ph.D Thesis, Federal University of São Carlos
Silva AA, Morabito R, Pureza V (2018) Optimization approaches to support the planning and analysis of travel itineraries. Expert Syst Appl 112(1):321–330
Spears WM, Jong KAD (1991) On the virtues of parameterized uniform crossover. In: Proceedings of the fourth international conference on genetic algorithms. Morgan Kaufman, San Mateo, pp 230–236
TSPLib Homepage. https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. Accessed 7 Jan 2019
TSP Test Data Homepage, http://www.math.uwaterloo.ca/tsp/data/index.html. Accessed 7 Jan 2019
Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83