A biased random-key genetic algorithm for the two-stage capacitated facility location problem

Fabrício Lacerda Biajoli
Antônio Augusto Chaves
Luiz Antonio Nogueira Lorena
Univ. Fed. de São Paulo*, São José dos Campos, SP, Brazil

Abstract

This paper presents a new metaheuristic approach for the two-stage capacitated facility location problem (TSCFLP), which the objective is to minimize the operation costs of the underlying two-stage transportation system, satisfying demand and capacity constraints. In this problem, a single product must be transported from a set of plants to meet customers demands passing out by intermediate depots. Since this problem is known to be NP-hard, approximated methods become an efficient alternative to solve real-industry problems. As far as we know, the TSCFLP is being solved in most cases by hybrid approaches supported by an exact method, and sometimes a commercial solver is used for this purpose. Bearing this in mind, a BRKGA metaheuristic and a new local search for TSCFLP are proposed. It is the first time that BRKGA had been applied to this problem and the computational results show the competitiveness of the approach developed in terms of quality of the solutions and required computational time when compared with those obtained by state-of-the-art heuristics. The approach proposed can be easily coupled in intelligent sys-

*MeMO-2020

tems to help organizations enhance competitiveness by optimally placing facilities in order to minimize operational costs.

Keywords: Two-stage capacitated facility location, Biased random-key genetic algorithm, Local search, Transportation systems.

1 Introduction

Facility location problems have numerous practical applications and have been studied extensively in the most active fields in Operations Research. In industrial engineering, the location analysis is one of the most important challenges, since it deals with the decision of optimally placing facilities in order to minimize operational costs. Decisions about facility location are strategic and belong to the core of any planning and management process, regardless of the industry involved. These decisions can influence the relation between the industrial sectors, the supply chain network and their customers.

Although solved in several practical situations by intuitive methods, optimal facility location decisions usually demand more in-depth studies (Fernandes et al., 2014) due of its importance. Different models have been proposed to solve the facility location problem in many real-world applications. For example, the single-stage capacitated facility location problem have been successfully analyzed, as can be seen in the review publications by Klose and Drexl (2005).

In a classic facility location problem, management decision makers have to decide which sites should be chosen to establish new facilities from a set of available candidate sites, while constraints are met in order to minimize the total cost and to guarantee that the demands of all customers have to be met, the capacity limits of the suppliers and facilities must be respected, etc. The cost includes fixed costs to open facilities and variable costs associated with transportation.

The two-stage capacitated facility location problem (TSCFLP) is a variant of classic facility location problem and can be defined as follows: a single product manufactured in plants is delivered to depots, both having limited capacities.

Then the products storage in depots are delivered to customers to satisfy their demands. The use of the plants and depots apply a fixed cost, while transportation from the plants to the customers through the depots results in a variable cost, depending on the quantity transported. The objective is identify what plants and depots to use, as well as the product flows from the plants to the depots and then to the customers such that the demands are met at a minimum cost.

TSCFLP has attracted researchers' attention, since in many situations more than one type of facilities are considered simultaneously. Due to the intractability of the TSCFLP, it is hard to obtain optimal solutions for a large-sized instance. Therefore, some researchers focused on heuristics rather than mathematical modeling in solving the TSCFLP (Guo et al., 2017).

Although the TSCFLP has some variants (e.g. Keskin and Üster, 2007; Klose, 2000; Tragantalerngsak et al., 2000), this paper will focus on the same version as presented by Litvinchev and Ozuna (2012), who proposed some Lagrangian relaxations for the problem. The Section 2 describes the mathematical model of the TSCFLP presented by the authors.

The same version is also considered in Fernandes et al. (2014) and Rabello et al. (2016). In the first paper a genetic algorithm (GA) was used to determine which plants and depots should be opened and obtained the flow values between plants and customers through the depots by solving a minimum cost flow problem. The authors also proposed two different constructive heuristics, one of them based on a pure greedy criterion and the other one on rounding linear relaxations of the problem formulation after iteratively fixing some of its variables. Rabello et al. (2016) proposed a hybrid method based on Simulated Annealing (SA) plus Clustering Search (CS) to determine the opened plants and depots, and a mathematical model to solve the transportation problem. The exact solver CPLEX v12.6 was used by them to obtain the solution of this mathematical model.

Recently, Guo et al. (2017) proposed a hybrid evolutionary algorithm framework with extreme machine learning fitness approximation for delivering solu-

tions. Genetic operators (i.e selection, crossover and mutation) are adopted to perform the search process as well as restarting strategy and a local search is used to refine the best solution found in the population. The authors also considered the same version as proposed by Litvinchev and Ozuna (2012).

This paper proposes a Biased Random-key Genetic Algorithm (BRKGA) to solve the TSCFLP. A local search for the transportation flow is also proposed in order to investigate and improve the best results found out by BRKGA. Computational results demonstrate that the BRKGA is competitive with state-of-the-art approaches (Fernandes et al., 2014; Rabello et al., 2016; Guo et al., 2017), providing better solutions for some instances developed by Fernandes et al. (2014).

This study contributes to the literature of TSCFLP by presenting an efficient approach to support and guide the process of placing facilities in order to minimize costs. In addition, it can help researchers, decision makers, developers of expert systems, and anyone else who requires these type of approaches. The main contributions are listed below:

- The literature about TSCFLP is limited and most approaches are supported by exact models that require a solver to execute. The proposed approach presents an independent and efficient method that can be easily coupled in expert systems.

- For the first time, the BRKGA is used to solve the TSCFLP. In particular, a simple and efective decoder is introduced for the problem and used as part of BRKGA metaheuristic.

- In addition, a new local search for TSCFLP is proposed. This method can be used to improve a TSCFLP solution generated by any heuristic approach.

The remainder of this paper is organized as follows. The Section 2 presents the mathematical formulation for the TSCFLP. In Section 3 the BRKGA algorithm is briefly introduced and in Section 4 the proposed BRKGA to solve

TSCFLP is described in detail. The computational experiments are presented in Section 5 followed by the conclusions and points out future research in Section 6.

2 Mathematical model for TSCFLP

To describe the problem, let I be the index set of potential plants, J the index set of potential depots and K the index set of customers. Let binary variables y_i ($i \in I$) and z_j ($j \in J$) which indicate, respectively, whether a plant or depot is opened (value equal to 1) or closed (value equal to 0). Let x_{ij} be a real variable which indicate the amount of products transported from plant $i \in I$ to depot $j \in J$. Similarly, let s_{jk} be a real variable which indicate the amount of products transported from depot $j \in J$ to customer $k \in K$. Finally, the parameters of this problem are defined as follows:

- f_i is the fixed cost associated to plant $i \in I$;
- g_j is the fixed cost associated to depot $j \in J$;
- c_{ij} is the transportation cost of one unit of the product between plant $i \in I$ and depot $j \in J$;
- d_{jk} is the transportation cost of one unit of the product between depot $j \in J$ and customer $k \in K$;
- q_k is the demand of customer $k \in K$;
- b_i is the capacity of plant $i \in I$; and
- p_j is the capacity of satellite $j \in J$.

Thus, the mathematical model of the TSCFLP can be formulated as:

$$Minimize \quad z = \sum_{i \in I} f_i y_i + \sum_{j \in J} g_j z_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} \sum_{k \in K} d_{jk} s_{jk} \qquad (1)$$

subject to:

$$\sum_{j \in J} s_{jk} \geq q_k, \qquad \forall k \in K \qquad (2)$$

$$\sum_{i \in I} x_{ij} \geq \sum_{k \in K} s_{jk}, \qquad \forall j \in J \qquad (3)$$

$$\sum_{j \in J} x_{ij} \leq b_i y_i, \qquad \forall i \in I \qquad (4)$$

$$\sum_{k \in K} s_{jk} \leq p_j z_j, \qquad \forall j \in J \qquad (5)$$

$$y_i \in \{0, 1\}, \qquad \forall i \in I \qquad (6)$$

$$z_j \in \{0, 1\}, \qquad \forall j \in J \qquad (7)$$

$$x_{ij} \in \mathbb{R}^+, \qquad \forall i \in I, \forall j \in J \qquad (8)$$

$$s_{jk} \in \mathbb{R}^+, \qquad \forall j \in J, \forall k \in K \qquad (9)$$

The objective function (1) minimize the total fixed cost associated with opening plants and depots plus the cost associated with both transportation stages. Constraints (2) is the demand constraint (for each customer, the demand must be met), (3) are conservation constraints (the total amount of products transported from a depot must be at most the total transported to it from the plants), (4) and (5) represent capacity limits for plants and depots, respectively. Finally, constraints (6) and (7) are assigned to flow variables, and constraints (8) and (9) impose binary values for the respective variables.

3 Biased Random-key Genetic Algorithm

A biased random-key genetic algorithm (BRKGA) is a general search metaheuristic proposed in Gonçalves and Resende (2011) and based on random-key genetic algorithm (RKGA), which was first introduced by Bean (1994) for

solving combinatorial optimization problems involving sequencing. The chromosomes of a RKGA are represented as vectors of randomly generated real numbers in the interval [0, 1]. A deterministic algorithm, called decoder, transforms a solution vector in a solution of the combinatorial optimization problem for which an objective value or fitness can be computed.

In a RKGA the population of random-key vectors is evolved over a number of iterations (generations). The initial population is made up of p vectors of random-keys. Each component of the solution vector is generated independently at random in the real interval $[0, 1]$. In each generation k, the decoder calculates the fitness of all individuals. Then the population is partitioned into two groups: a small group of elite individuals (p_e, i.e. those with the best fitness values) and the remaining set of non-elite individuals ($p - p_e$).

A new generation of individuals is produced by copying all elite individual of the population of generation k without modification to the population of generation $k + 1$. The mutation process of RKGAs is done by introducing mutants into the population. A mutant is simply a vector of random keys generated in the same way that an element of the initial population is generated. At each generation, a small number (p_m) of mutants is introduced into the population. With the p_e elite individuals and the p_m mutants accounted for in population $k + 1$, $p - p_e - p_m$ additional individuals need to be produced to complete the p individuals that make up the new population. This is done by producing $p - p_e - p_m$ offspring through the process of mating or crossover.

A BRKGA differs from a RKGA in the way parents are selected for mating. While a RKGA selects two parents at random from the entire population, in a BRKGA one parent is selected at random from the elite partition in the current population and one from the non-elite partition. In both algorithms, the mating is done by parameterized uniform crossover, where an offspring inherits the vector component of its elite parent with probability $\rho_e > 0.5$ and of its non-elite parent with probability $1 - \rho_e$.

When the next population is complete, fitness values are computed by the decoder for all of the newly created random-key vectors and the population is

partitioned into elite and non-elite individuals to start a new generation.

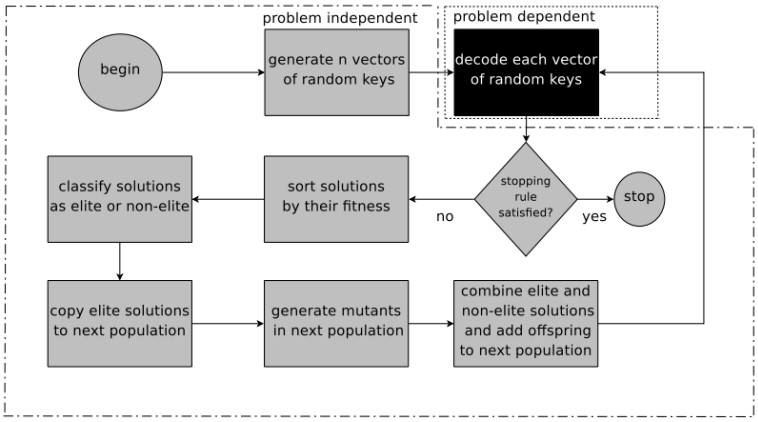


Figure 1: Flowchart of a BRKGA (Source: Gonçalves and Resende, 2011)

Figure 1 shows a flow diagram of the BRKGA framework with a clear separation between the problem dependent and problem independent components of the method (Gonçalves and Resende, 2011). To describe a BRKGA for a specific combinatorial optimization problem, one need only to show how solutions are encoded as vectors of random keys and how these vectors are decoded to feasible solutions of the optimization problem (problem dependent). In the next section, the BRKGA applied for TSCFLP is described.

4 BRKGA for the two-stage capacitated facility location problem

This section shows how solutions are encoded to a vector of random keys and how they are decoded from a vector of random keys into a TSCFLP solution.

4.1 Encoding a solution to a vector of random keys

A solution is encoded as a vector $V = (v_1, ..., v_n)$ of size $n = |I| + |J| + |K|$, where $|I|$ is the number of plants, $|J|$ is the number of depots, $|K|$ is the number

of customers and v_x is a random number in the interval [0, 1], for $x = 1, ..., n$. A simple example of a solution encoded as a vector is shown in Figure 2, with three plants (p_i, for $i = 1, ..., 3$), four depots (d_j, for $j = 1, ..., 4$) and five customers (c_k, for $k = 1, ..., 5$).

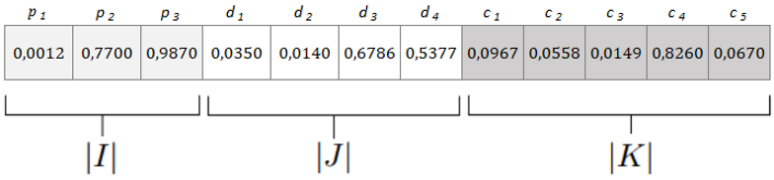


Figure 2: Example of a solution encoded as a vector

4.2 Decoding a solution from a vector of random keys

The decoding process of a chromosome into a TSCFLP solution proposed in this paper consists of four steps: select plants and depots; verify if the customers demands is met; remove superfluous facilities; and calculate the transportation flow in two stages (from plants to depots and from depots to customers). Figure 3 illustrates the sequence of steps applied to each chromosome in the decoding process.

In the first step, the decoder takes as input a chromosome, i.e. the vector of random keys. A partial solution with plants and depots opened (selected) is produced by first sorting the vector in descending order to produce a sorted list of facilities. For each position of the vector, the plants and depots are selected if the random key value is equal or greater than 0.5. The Figure 4 illustrates an example in which the plants p_3 and p_2 as well as depots d_3 and d_4 are selected in the vector, respecting the order.

Note that, when the vector is sorted the order of the facilities changes, producing different solutions for different values of random keys.

The second step verifies if the plants and depots capacities meet the customers demands. If the demand is satisfied, then the second step is not executed. If the demand is not met, then plants, depots or both will be selected

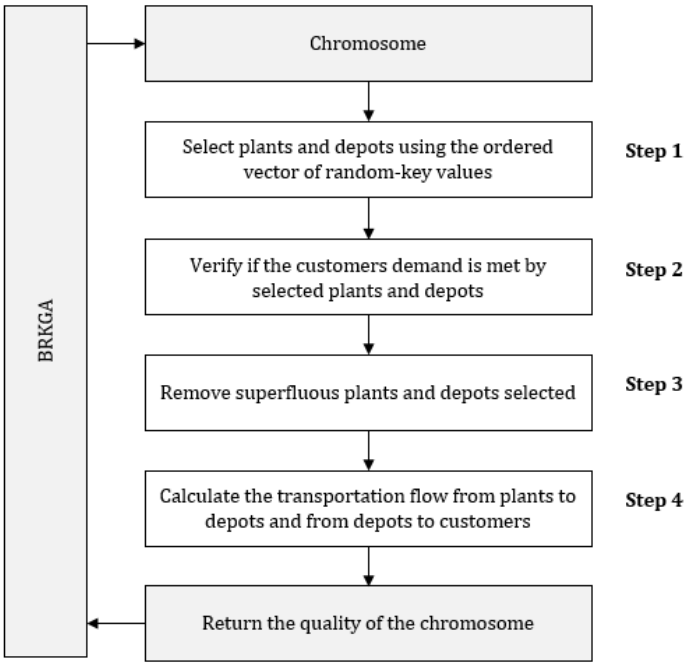


Figure 3: Sequence of steps applied to each chromosome in the decoding process

p_3	c_4	p_2	d_3	d_4	c_1	c_5	c_2	d_1	c_3	d_2	p_1
0,9870	0,8260	**0,7700**	**0,6786**	**0,5377**	0,0967	0,0670	0,0558	0,0350	0,0149	0,0140	0,0012

Figure 4: Example of a random-key vector sorted

until customers' demand can be met. This selection is not aleatory, it respects the order of a facility in the random-key vector. For example, for plants, the algorithm selects the first unselected plant in the vector, analyzes the new total capacity and stops if the customers demands are met. Otherwise, the next plant is selected. For depots the process is the same. At the end of this step, a feasible solution was builded.

The third step of the decoder attempts to remove superfluous plants and superfluous depots from solution. Basically, if the total capacity of plants is greater than the customers demands, then maybe it is possible to remove one or more plants from the solution. While there is some selected plant such that the

solution is still feasible, then such plant having the smallest index is removed from the solution. The process for depots works along the same lines.

The second and third steps of the decoder not only return a feasible selection of plants and depots, but also modify the vector of random keys V such that it decodes directly into the same solution with the application of only the first step. To do this the decoder reset V as follows:

$$v_x = 1 - v_x$$

where x is the position of a selected facility in the second step or the position of a removed facility it the third step.

In the forth and last step the decoder defines the transportation flow from plants to depots and from depots to customers. For each plant the decoder builds a list of opened depots sorted by the transportation costs in increasing order. Then it selects the first depot, transports the maximum quantity and selects another one until the plant capacity is over or the total customer demand is met. The same process is performed for the flow from depots to customers. For each depot the decoder builds a list of customers sorted by the transportation costs, also in increasing order. Then it selects the first customer, transports the maximum quantity to customer and selects the next customer from the list. The process stops when the depot capacity is over or the total customer demand is met.

4.3 Local search

The best solution generated by BRKGA is not necessarily optimal and often may be improved by local search heuristics. This paper proposes a local search for the transportation flow of TSCFLP. This local search is applied in the ten best solutions with distinct objective functions value generated by BRKGA at the end of its execution.

The local search performs moves by swapping the quantity transported in two steps: first from a plant to a depot and second from a depot to a customer. These moves are made by taking two plants and one depot from each

transportation flow. Then the minimum quantity transported is swapped from plants. For example, the plant p_1 transports 100 units of a product to the depot d_1 and the plant p_2 transports 150 units to depot d_2. As the result of a move, the plant p_1 transports 100 units to d_2 and the plant p_2 transports 100 units to d_1 and 50 to d_2.

The neighborhood search moves are performed iteratively for each plant and for each depot in the current solution. Only the best move is made and if an improving neighbor solution is found, it becomes the new current solution. Then the local search is restarted from it. The process is repeated until it is impossible to find a better solution in the neighborhood.

In the second step of the method the same process is performed, however, it considers the transportation flow from depots to customers.

4.4 Parameter setting

The problem independent module of a BRKGA has parameters that need to be set, listed as follows:

- n: number of genes in a chromosome;
- p: population size;
- p_e: size of elite solution population;
- p_m: size of the mutant solution population; and
- ρ_e: probability that the gene of the offspring inherits the allele of the elite parent.

In Gonçalves and Resende (2011), the authors define experimentally a range of values that can be used for these parameters, required for using the framework. However, since it's a challenging and time consuming task to tune the appropriate value of the parameters even with a pre-established range, this paper used the CALIBRA procedure (an automatic tuning procedure by Adenso-Díaz

and Laguna, 2006) to effectively determine the value of the parameters p_e, p_m and ρ_e. The value of the other two parameters (n and p) was fixed.

Table 1 presents the parameter setting used in this paper, which is in compliance with the range defined by the authors of BRKGA (Gonçalves and Resende, 2011), since this range was used as the CALIBRA procedure range. Experimental results obtained in test instances show that the parameters values tuned by CALIBRA outperforms those values empirically chosen in the quality of solution and execution time.

Table 1: Parameter setting of BRKGA

Parameter	Description	Value
n	size of chromosome	$\|I\| + \|J\| + \|K\|$
p	size of population	$4n$
p_e	size of elite population	$0.13p$
p_m	size of mutant population	$0.17p$
ρ_e	elite allele inheritance probability	0.69

4.5 Implementation details

The algorithm described in this paper was implemented using the BRKGA framework, a C++ framework for BRKGA (Toso and Resende, 2015).

As described in Section 3, a BRKGA consists of a problem dependent phase (the decoder) and a general-purpose problem independent phase (Figure 1). Besides that, the BRKGA framework was designed as an object-oriented, multithreading, general-purpose framework which implements all problem independent components and provides a simple way for chromosome decoding. For that reason, this framework enables seamless interaction with any problem-specific decoder.

The algorithm proposed uses OpenMP, a multi-platform application programming interface (API) for shared-memory parallel programming in C, C++, and FORTRAN. In the computational experiments, this paper makes use of a

parallel processing, setting to use two parallel threads in chromosome decoding.

The algorithm runs for three independent populations, makes use of periodic exchanges of elite individuals (at every 100 generations the two best individuals are exchanged) and re-initialization strategy, i.e. restarts the algorithm with new random keys if it runs 200 generations without improving the best solution.

5 Computational experiments

The objective of this section is to evaluate the performance of BRKGA+LS (BRKGA + Local Search) proposed in this paper. The algorithm was implemented in C++ and all the computational experiments were carried out in a computer with an Intel i5 1.7GHz processor and 16GB of RAM under a Windows 10 environment. The following describes the computers used by literature referenced in the comparisons of this paper. Fernandes et al. (2014) performed the computational experiments on a Pentium Intel with 2.3GHz processor and 24GB of RAM. In Rabello et al. (2016), the experiments was performed on Intel i5 2.67GHz processor and 4GB of RAM. Guo et al. (2017) conducted the experiments on a personal computer with an Intel i7 3.6GHz processor and 8GB of RAM.

A set of instances developed by Fernandes et al. (2014) was used in the experiments. The benchmark instances consists of two sets with 25 instances each one. In the first set, the number of plants, depots and customers are 50, 100 and 200, respectively. In the second set, these values are twice as much, i.e., 100 plants, 200 depots and 400 customers. Like in Fernandes et al. (2014), the number of plants is used to indicate the size of the considered instances.

For each instance the algorithm was running 10 times, using a different random number generator seed for each run. The best objective function value and the average objective value are recorded by the algorithm. Like in Guo et al. (2017), the performance of each algorithm is measured by the relative percentage deviation (RPD) defined by the equation

$$RPD(\%) = \frac{(Z(alg) - Z(LB)) \times 100}{Z(LB)}$$

where $Z(alg)$ is the solution value delivered by a specific algorithm and $Z(LB)$ is the lower bound of the corresponding instance (best solutions obtained by the exact solver Gurobi v6.5 and presented in Guo et al., 2017).

Table 2 report the computational results for the instances with 50 plants, obtained by BRKGA+LS and by the state-of-the-art algorithms. Table 3 shows the same comparison for the instances with 100 plants. In both tables, the first column refers to the class of problems and the second one indicates the identification of the instance. The other ones show the lower bound (LB), the RPD calculated for the average of the solutions (RPD_a), the RPD of the best solution (RPD_b) and the average execution time (in seconds) of the algorithms. The value of RPD_b is not presented for Fernandes et al. (2014), because the authors presented only the average value of their solutions. The average value of the measures is presented at the end of the tables. The boldface font represents the best value for the instances.

Table 2: Computational results for instances with 50 plants.

			GA[1]		CS+CPLEX[2]			HEA/FA[3]			BRKGA+LS		
Class	Id	LB	RPD_a (%)	Time(s)	RPD_b (%)	RPD_a (%)	Time(s)	RPD_b (%)	RPD_a (%)	Time(s)	RPD_b (%)	RPD_a (%)	Time(s)
	1	721209.6	**0.13**	581.41	**0.13**	0.31	69.92	**0.13**	0.24	416.17	0.14	0.14	23.33
	2	730451.6	0.40	564.71	**0.24**	0.28	140.93	0.31	0.36	492.78	**0.24**	**0.24**	140.00
1	3	731885.3	0.24	578.28	**0.22**	0.28	108.82	**0.22**	0.24	316.08	**0.22**	**0.22**	47.45
	4	721515	0.81	533.44	1.19	1.33	175.91	0.54	0.59	455.55	**0.50**	**0.51**	140.44
	5	713633.8	0.82	552.90	**0.81**	0.84	62.16	0.86	0.97	463.69	**0.81**	**0.81**	71.37
	1	479860.2	**2.69**	317.05	**2.69**	2.71	44.54	2.74	2.82	230.84	**2.69**	**2.69**	20.34
	2	483072.2	**2.30**	316.55	**2.30**	2.42	95.77	2.34	2.41	199.11	**2.30**	2.31	73.16
2	3	486018.5	2.14	330.70	**1.87**	**1.87**	35.55	**1.87**	1.89	167.28	**1.87**	**1.87**	141.54
	4	482374.6	2.04	312.35	**2.02**	**2.03**	115.23	2.07	2.12	162.13	**2.02**	2.05	71.96
	5	474803.3	3.14	276.85	**3.12**	3.23	75.96	**3.12**	**3.12**	170.57	**3.12**	**3.12**	13.09
	1	2608800	**3.07**	276.45	**3.07**	**3.07**	43.61	3.14	3.30	117.09	3.11	3.11	153.30
	2	2616252	3.12	285.95	**3.11**	**3.11**	57.57	3.30	3.36	116.67	3.17	3.17	48.62
3	3	2598277	3.11	271.31	**3.10**	**3.10**	64.78	3.30	3.39	161.80	**3.10**	**3.10**	88.09
	4	2612534	3.07	236.55	**3.06**	**3.06**	77.92	3.18	3.22	148.86	3.10	3.10	9.96
	5	2568856	**3.01**	242.34	**3.01**	**3.01**	37.27	3.17	3.19	158.15	3.13	3.13	48.58
	1	525294.1	**3.14**	303.57	**3.14**	3.48	115.50	3.36	3.54	216.81	**3.14**	3.22	70.58
	2	526911.7	**2.33**	307.04	**2.43**	2.50	107.58	2.74	2.92	199.39	**2.43**	2.51	97.80
4	3	532592.3	2.66	318.14	2.45	2.45	148.90	2.59	2.62	247.29	**2.33**	**2.42**	133.19
	4	529372	2.53	279.54	**2.36**	**2.48**	156.22	2.63	2.81	261.75	2.44	2.51	90.39
	5	521470.1	**3.13**	264.10	**3.13**	3.29	89.83	3.18	3.18	182.24	3.14	3.16	47.79
	1	2743547	1.20	361.04	1.18	**1.19**	193.13	**1.16**	1.20	199.23	1.18	1.22	145.41
	2	2752021	**1.07**	344.12	**1.07**	1.10	160.30	1.11	1.16	259.72	**1.07**	1.09	43.13
5	3	2737769	1.10	420.73	**1.09**	**1.09**	235.62	1.22	1.28	202.71	1.13	1.15	157.52
	4	2748216	**1.07**	300.55	**1.06**	1.11	234.59	1.10	1.13	203.32	1.10	1.10	183.09
	5	2702350	**1.25**	318.55	**1.23**	**1.25**	62.26	1.31	1.34	182.20	1.26	1.27	89.61
		Average	1.98	355.77	1.96	2.02	108.39	2.03	2.10	237.26	**1.95**	**1.97**	**85.99**

[1] Fernandes et al. (2014); [2] Rabello et al. (2016); [3] Guo et al. (2017).

Table 3: Computational results for instances with 100 plants.

			GA[1]		CS+CPLEX[2]			HEA/FA[3]			BRKGA+LS		
Class	Id	LB	RPD_a (%)	Time(s)	RPD_b (%)	RPD_a (%)	Time(s)	RPD_b (%)	RPD_a (%)	Time(s)	RPD_b (%)	RPD_a (%)	Time(s)
	1	1475952	0.55	2784.60	0.11	0.18	400.20	0.29	0.33	1341.67	**0.10**	**0.11**	320.99
	2	1462736	1.01	2745.02	0.37	0.68	655.95	0.27	0.43	1514.37	**0.12**	**0.13**	661.34
1	3	1492163	0.34	3001.83	0.70	0.91	523.17	0.23	0.48	1481.83	**0.15**	**0.17**	232.45
	4	1459076	0.49	2823.39	**0.22**	0.30	740.68	0.25	0.28	1122.33	**0.22**	**0.23**	344.14
	5	1490742	0.67	2863.24	**0.12**	0.42	409.80	0.17	0.24	1501.11	**0.12**	**0.12**	622.30
	1	970908.5	0.89	1483.25	**0.27**	0.39	884.59	0.63	0.70	944.83	**0.27**	**0.28**	481.18
	2	965908.5	0.74	1455.77	**0.28**	0.46	611.62	0.47	0.56	908.30	0.29	**0.29**	553.57
2	3	975499.7	1.42	1427.72	**0.14**	0.20	236.58	0.20	0.25	922.00	**0.14**	**0.15**	108.15
	4	973019.1	0.56	1444.07	**0.28**	0.38	715.60	0.56	0.59	1083.61	**0.28**	**0.29**	717.48
	5	941567	1.12	1419.02	**0.60**	0.68	427.82	0.86	0.96	1159.33	0.61	**0.61**	612.25
	1	5213566	1.63	1355.38	**1.61**	**1.62**	1049.04	1.91	2.03	855.68	1.64	1.65	851.26
	2	5191321	1.67	1320.83	**1.65**	**1.65**	1217.64	1.96	1.99	1220.48	1.68	1.71	283.62
3	3	5145991	**1.58**	1311.86	**1.58**	**1.58**	989.07	1.78	1.84	1193.12	1.63	1.63	109.46
	4	5225601	1.74	1365.90	**1.71**	**1.73**	1469.17	1.98	2.01	968.64	1.74	1.75	378.38
	5	5163182	1.72	1383.04	**1.68**	**1.69**	1404.04	1.99	2.08	942.29	1.72	1.75	495.11
	1	1052172	0.82	1269.01	**0.60**	0.76	807.73	0.91	1.04	878.64	0.64	**0.69**	528.51
	2	1043553	0.93	1230.08	**0.68**	0.83	462.41	0.82	0.89	672.60	**0.68**	**0.74**	600.17
4	3	1050683	1.88	1283.26	**0.63**	**0.80**	734.64	1.00	1.29	877.26	0.73	1.04	683.77
	4	1044571	0.96	1301.32	**0.75**	**0.85**	749.02	1.36	1.62	797.26	0.81	0.88	692.03
	5	1053869	0.64	1334.96	**0.53**	**0.73**	381.18	0.94	1.06	843.02	0.58	0.61	597.65
	1	5486098	0.48	1551.16	**0.39**	**0.40**	873.59	0.61	0.62	967.57	**0.39**	0.41	433.52
	2	5461680	0.47	1499.94	**0.39**	**0.42**	336.05	0.41	0.44	1017.10	0.41	0.43	604.48
5	3	5425391	0.62	1477.01	0.50	0.52	478.28	0.64	0.73	1371.96	**0.42**	**0.45**	671.41
	4	5494811	0.52	1513.78	**0.41**	**0.44**	436.70	0.58	0.58	1084.43	0.43	**0.44**	438.47
	5	5442621	0.47	1472.05	**0.39**	**0.40**	1080.57	0.43	0.47	1162.63	**0.39**	**0.40**	607.44
Average			0.96	1625.37	0.66	0.76	723.01	0.85	0.94	1073.28	**0.65**	**0.68**	**505.17**

[1] Fernandes et al. (2014); [2] Rabello et al. (2016); [3] Guo et al. (2017).

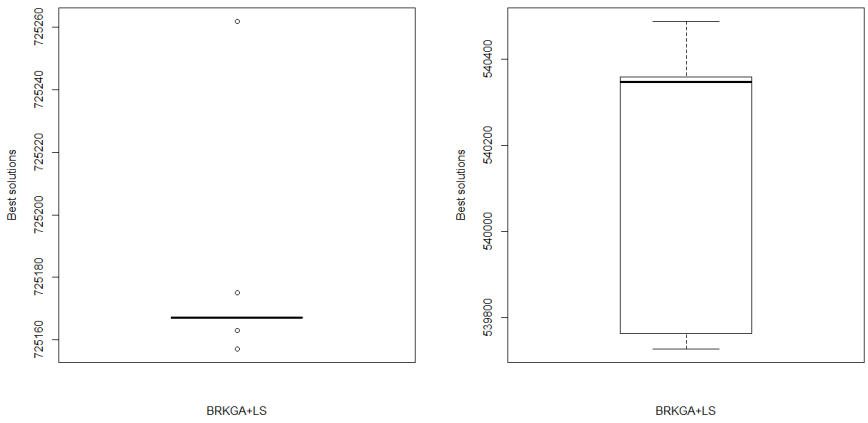
Some conclusions can be derived from the results presented by Table 2 and Table 3. In 29 instances the average results (RPD_a) were improved by BRKGA+LS when compared with GA (Fernandes et al., 2014), especially in instances with 100 plants. In the other ones, the algorithm gives good solutions in a reasonable computational time. When compared with HEA/FA (Guo et al., 2017), BRKGA+LS outperformed the best results (RPD_b) in 44 instances and achieve the same result in 4 instances. In 10% of instances of class 3, BRKGA+LS obtained the same solutions than the others algorithms. The main feature of the instances in class 3 is the large capacity of its plants and depots, which results in a lesser amount of opened facilities in the optimal solution (Fernandes et al., 2014).

Tables 2 and 3 also shows the comparison between BRKGA+LS and CS+CPLEX proposed by Rabello et al. (2016), that used an exact solver as part of the method. The results presented demonstrate that the proposed approach outperformed the best results of state-of-the-art CS+CPLEX in 6 instances and achieved the same RPD in other 20 instances. In some others CS+CPLEX presenting the best known solutions of the literature. Despite this, the solutions found by the proposed algorithm are competitive due the low difference in RPD presented in all instances. Moreover, BRKGA+LS outperformed CS+CPLEX in most instances when compared the average solutions, demonstrating the robustness of the BRKGA approach.

Box plots illustrating the solution quality and robustness of the proposed method are shown in Figure 5. The rectangles represent the middle half of the solutions for each instance, between the lower quartile (25%) and the upper quartile (75%). The lines go from the minimum to the maximum solutions, and the middle line represents the median solution. The open circles outside the lines show the outliers.

The most of the instances presented the same behavior as the box plot (a) and (d).

Furthermore, BRKGA+LS outperformed the other approaches in average of RPD_a, RPD_b and execution time for both sets of instances (see the last line



(a) *Instance Class 1, Id 4 with 50 plants* (b) *Instance Class 4, Id 3 with 50 plants*

(c) *Instance Class 1, Id 1 with 100 plants* (d) *Instance Class 5, Id 3 with 100 plants*

Figure 5: Box plot of some solutions found by BRKGA+LS.

presented in the Tables 2 and 3). Remarkably, the BRKGA+LS demands lesser time do find solutions. For example, Figure 6 illustrates run-time distributions, or time-to-target (TTT) plot (Aiex et al., 2007), for instance Class 1, Id 4 with 50 plants (see Table 2), which presented the greatest improvement in the set of tests. The experiment consists in running the BRKGA+LS 100 times. Each run is independent of the other and stops when a solution with a cost which is at least as good as a given target value (Rabello et al., 2016) is found. The probability of BRKGA+LS to find a solution better than the target value in at most seven seconds is about 50%, in at most eight seconds is about 80% and in at most nine seconds is about 99%. Table 2 shows that the target value (Rabello et al., 2016) for the same instance was found in 175.91 seconds (the difference between the computers used is not significant).

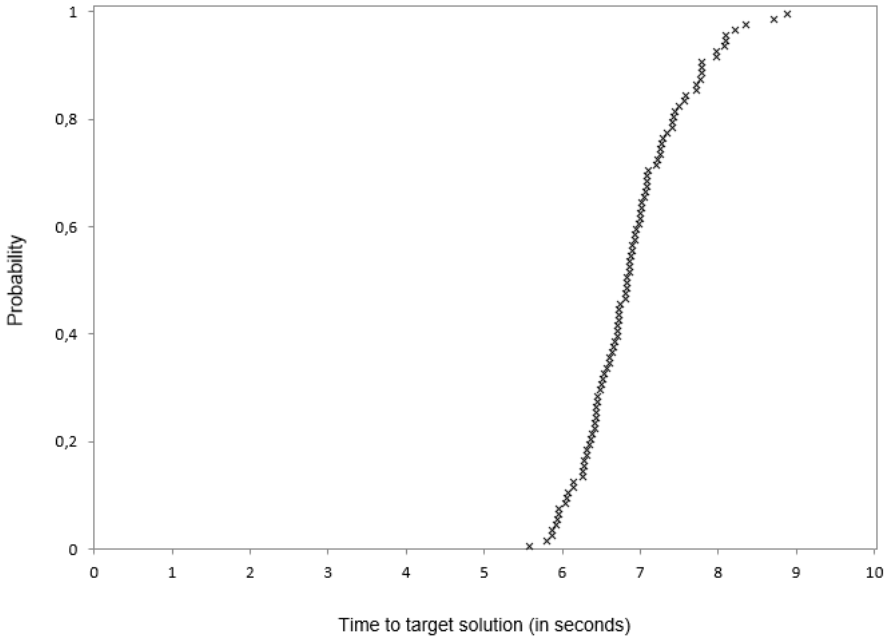


Figure 6: Time in seconds to target distributions of BRKGA+LS for instance Class 1, Id 4 with 50 plants.

6 Conclusion

This paper presents a biased random-key genetic algorithm (BRKGA) which uses a decoder with four phases for the two-stage capacitated facility location problem (TSCFLP). A literature review showed that BRKGA had never been applied to this problem, which has attracted researchers' attention in last few years. A local search (LS) that performs exchanges in transportation flow was also presented.

The performance of the proposed BRKGA+LS was tested and evaluated on two sets of benchmark instances. The computational experiments showed that, in general, the approach found good solutions for the TSCFLP in an acceptable time for all instances. When compared with approaches that did not use an exact solver, BRKGA improved the solutions in most of the instances. In the other instances the difference between the RPD of the average solutions (RPD_a) and RPD of the best solutions (RPD_b) is minimum. In comparison with an approach that use an exact solver to performs the transportation flow, the proposed solution presented best results in 11% of the instances. On the other hand, the average of the solutions and the computational execution time are improved in most instances.

Although the approach with BRKGA has some strengths such as flexibility, simplicity, parallel processing, generic and simple structure that requires only a specific decoder for a specific problem and a freely-available application programming interface (API) that can be used as basis for the implementation (Toso and Resende, 2015), the parameter calibration process required many tests and hours of preprocessing using the CALIBRA procedure (Adenso-Díaz and Laguna, 2006). However, the parameter selection can significantly affect the quality of the algorithm in terms of solution performance and computational time, and therefore it is worthwhile to use a method to calibrate the parameters, even if it requires long computational time once.

After all, BRKGA with the local search proposed demonstrates that it is a simple but effective approach, capable of returning good solutions very quickly

for the TSCFLP. The approach also demonstrates that, with few modifications, it can be used to solve other facility location problems. In addition, the approach can be easily coupled in any expert system to generate solutions in accordance with the inputs. Naturally, its parameters need to be calibrated by the system according to the inputs and solutions being produced. However, the decision maker may help the system setting new values for the parameters as necessary.

Further research includes the consideration of a new decoder in association with a local search in order to investigate solutions beyond the guided solution generated by the deterministic decoding process. In addition, applying the proposed local search during the evolutionary process of BRKGA can be done to explore a larger solution space. The use of another automated methods to calibrate the parameters of BRKGA can also be studied and tested, such as the use of an adaptive version of BRKGA that calibrate the parameters automatically during the evolutionary process.

References

References

Adenso-Díaz, B., Laguna, M., 2006. Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research 54, 99–114. doi:10.1287/opre.1050.0243.

Aiex, R., Resende, M., Ribeiro, C., 2007. TTT plots: a perl program to create time-to-target plots. Optimization Letters 1, 355–366.

Bean, J.C., 1994. Genetic algorithms and random keys for sequencing and optimization. INFORMS Journal on Computing 6, 154–160.

Fernandes, D.R.M., Rocha, C.T.M., Aloise, D.J., Ribeiro, G.M., Santos, E.M., Silva, A., 2014. A simple and effective genetic algorithm for the two-stage capacitated facility location problem. Computers & Industrial Engineering 75, 200–208.

Gonçalves, J.F., Resende, M.G.C., 2011. Biased random-key genetic algorithms for combinatorial optimization. J. Heuristics 17, 487–525.

Guo, P., Cheng, W., Wang, Y., 2017. Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems. Expert Systems With Applications 71, 57–68. doi:10.1016/j.eswa.2016.11.025.

Keskin, B.B., Üster, H., 2007. A scatter search-based heuristic to locate capacitated transshipment points. Computers & OR 34, 3112–3125. doi:10.1016/j.cor.2005.11.020.

Klose, A., 2000. A lagrangian relax-and-cut approach for the two-stage capacitated. European Journal of Operational Research 126, 185—198.

Klose, A., Drexl, A., 2005. Facility location models for distribution system design. European Journal of Operational Research 162, 4–29.

Litvinchev, I., Ozuna, E.L., 2012. Lagrangian bounds and a heuristic for the two-stage capacitated facility location problem. International Journal of Energy Optimization and Engineering (IJEOE) 1, 59–71.

Rabello, R.L., Mauri, G., Ribeiro, G.M., 2016. Método heurístico híbrido para resolućão do problema de localizaćão de facilidades capacitadas em dois níveis, in: Anais do XLVIII SBPO - Simpósio Brasileiro de Pesquisa Operacional, Vitória, ES - Brasil. pp. 2460–2471. URL: http://www.din.uem.br/sbpo/sbpo2016/pdf/156298.pdf.

Toso, R.F., Resende, M.G.C., 2015. A c++ application programming interface for biased random-key genetic algorithms. Optimization Methods and Software 30, 81–93.

Tragantalerngsak, S., Holt, J., Ronnqvist, M., 2000. An exact method for the two-echelon, single-source, capacitated facility location problem. European Journal of Operational Research 123, 473–489.