Information Processing Letters 116 (2016) 513–520

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A Hybrid Biased Random Key Genetic Algorithm for the Quadratic Assignment Problem

Eduardo Lalla-Ruiz *, Christopher Expósito-Izquierdo, Belén Melián-Batista, J. Marcos Moreno-Vega

Department of Computer and Systems Engineering, Universidad de La Laguna, Spain

ARTICLE INFO

Article history:
Received 12 September 2014
Received in revised form 17 January 2016
Accepted 3 March 2016
Available online 18 March 2016
Communicated by X. Wu

Keywords:
Quadratic Assignment Problem
Biased Random Key Genetic Algorithm
Metaheuristic
Approximation algorithms

ABSTRACT

The Quadratic Assignment Problem (QAP) is a well-known $\mathcal{NP}$-hard combinatorial optimization problem that has received a lot of attention from the research community since it has many practical applications, such as allocation of facilities, design of electronic devices, etc. In this paper, we propose a hybrid approximate approach for the QAP based upon the framework of the Biased Random Key Genetic Algorithm. This hybrid approach includes an improvement method to be applied over the best individuals of the population in order to exploit the promising regions found in the search space. In the computational experiments, we evaluate the performance of our approach on widely known instances from the literature. In these experiments, we compare our approach against the best proposals from the related literature and we conclude that our approach is able to report high-quality solutions by means of short computational times.

1. Introduction

The Quadratic Assignment Problem (QAP) is a combinatorial optimization problem introduced by Koopmans and Beckman [14]. Input data for the QAP are a set of facilities denoted as $\mathcal{F} = \{1, 2, \ldots, n\}$ and a set of locations denoted as $\mathcal{L} = \{1, 2, \ldots, n\}$. Each pair of facilities, $(i, j) \in \mathcal{F}$, requires a certain flow, denoted as $f_{ij} \geq 0$. The distance between the locations $k, l \in \mathcal{L}$ is denoted as $d_{kl} \geq 0$. It should be mentioned that the flows and distances are symmetric (*i.e.*, $f_{ij} = f_{ji}, \forall i, j \in \mathcal{F}$ and $d_{kl} = d_{lk}, \forall k, l \in \mathcal{L}$) and the flow/distance between a given facility/location and itself is zero (*i.e.*, $f_{ii} = 0, \forall i \in \mathcal{F}$ and $d_{kk} = 0, \forall k \in \mathcal{L}$).

The objective of the QAP is to minimize the cost derived from the distance and flows among facilities. This can be formally expressed as minimizing the following expression:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)}, \tag{1}$$

where ϕ is a solution belonging to the set composed of all the feasible permutations, denoted as $\mathcal{S}_n$, such that $\phi : \mathcal{F} \rightarrow \mathcal{L}$. The cost associated to assign facility i to location $\phi(i)$ and facility j to facility $\phi(j)$ is, according to Equation (1), $f_{ij} d_{\phi(i)\phi(j)}$. In addition, let us denote as $f(\phi)$ the objective function value of solution $\phi \in \mathcal{S}_n$. A comprehensive description of the QAP is provided by Burkard et al. [2].

The QAP is known to belong to the $\mathcal{NP}$-hard class (Sahni and Gonzalez [17]). In fact, there are no exact methods in the literature which can tackle the QAP in medium scenarios ($n > 25$) by means of reasonable computational times. Nowadays, its hardness and heterogeneous applications turn the QAP a challenging problem within the

* Corresponding author.
E-mail addresses: elalla@ull.es (E. Lalla-Ruiz), cexposit@ull.es (C. Expósito-Izquierdo), mbmelian@ull.es (B. Melián-Batista), jmmoreno@ull.es (J.M. Moreno-Vega).

http://dx.doi.org/10.1016/j.ipl.2016.03.002
0020-0190/

optimization field. Additionally, many well-known problems such as the Travelling Salesman Problem or Graph Partitioning can be formulated as the QAP. In this context, the QAP has served as proving ground for algorithmic proposals over the last decades. Exact methods have been proposed by Fedjki and Duffuaa [4] and Erdoğan and Tansel [10]. Drezner et al. [7] review the applicability of widespread metaheuristics from the literature to address the QAP. The interested reader is referred to the detailed survey provided by Loiola et al. [15].

There are many practical applications of the QAP in the literature. For instance, Duman and Or [8] discuss how to carry out the sequencing of placement and configuration of feeder in printed circuit boards. Cheng et al. [5] model the passenger walking distance in airports according to the passenger transfer volume between aircrafts and distance between gates. Finally, Wu et al. [20] describe an application within the field of coding theory.

The remainder of this paper is organized as follows. Section 2 describes the Hybrid Biased Random Key Genetic Algorithm proposed to address the QAP. Afterwards, Section 3 analyzes the performance of our proposal in realistic scenarios. Finally, Section 4 draws forth the main conclusions extracted from the work and suggests several directions for further research.

2. Hybrid Biased Random Key Genetic Algorithm

Genetic Algorithms (GAs) are bio-inspired algorithms based upon the concepts of biological evolution and survival of the fittest individuals (Holland [13]). One of the major drawbacks of GAs is the difficulty to maintain feasibility through successive generations. With the goal of avoiding this fact, Bean [1] introduced the concept of *random key*. A random key is a real-valued number defined in $[0, 1)$, whereas a random key vector is an element of the $[0, 1)^{\rho}$ space, where ρ depends on the dimension of the optimization problem at hand. For instance, $\rho = n$ when addressing the Quadratic Assignment Problem (QAP).

A Random Key Genetic Algorithm (RKGA) is a variant of GA in which the chromosomes are random key vectors. The reproduction is performed by copying a subset of elite individuals (*i.e.*, those individuals with the lowest objective function value) from the current population to the next one. In this case, the parameterized uniform crossover suggested by Spears and De Jong [18] is used as crossover strategy. This strategy involves tossing a biased coin for each gene in order to determine which parent contributes to the corresponding gene of the relevant offspring solution. Finally, a set of random individuals is included into the current population during the mutation phase.

A variation of RKGA was presented by Ericsson et al. [11], in which one parent is selected from the set of elite individuals and the other one from the rest of the population when applying the crossover operator. In this case, a biased coin favouring the elite parent is tossed during the crossover. Although this specialized version of the RKGA was proposed as a heuristic to solve a particular problem (*i.e.*, the Weight Setting Problem), it contained the germ of what in the subsequent paper by Gonçalves and Resende [12] would be identified as a general-purpose metaheuristic: the Biased Random Key Genetic Algorithm (BRKGA).

Algorithm 1: Hybrid Biased Random Key Genetic Algorithm.

Require: $\mathcal{G}$, number of generations
Require: t, size of the population
Require: e, number of elite individuals in the population
Require: m, number of mutant individuals in the population
Require: α, crossover rate
Require: n, number of facilities in the QAP
Ensure: Best solution found for the QAP
1: Create $\mathcal{P}(1)$ with random key vectors composed of n random keys by means of Solution Generator Procedure
2: Evaluate the fitness of each individual in $\mathcal{P}(1)$
3: **for** $(g = 2 \dots \mathcal{G})$ **do**
4: $\quad\mathcal{P}(g) = \mathcal{P}_e(g-1)$
5: $\quad$Apply improvement method over each individual included into $\mathcal{P}(g)$
6: $\quad$Include m mutant individuals in $\mathcal{P}(g)$ with Solution Generator Procedure
7: $\quad$**while** $(|\mathcal{P}(g)| \leq t)$ **do**
8: $\quad\quad rk_1 \leftarrow$ Select an individual at random from $\mathcal{P}_e(g-1)$
9: $\quad\quad rk_2 \leftarrow$ Select an individual at random from $\mathcal{P}(g-1) \setminus \mathcal{P}_e(g-1)$
10: $\quad\quad rk \leftarrow$ Crossover(rk_1, rk_2, α)
11: $\quad\quad\mathcal{P}(g) = \mathcal{P}(g) \cup \{rk\}$;
12: $\quad$**end while**
13: $\quad$Evaluate the fitness of each individual in $\mathcal{P}(g)$
14: **end for**
15: **return** Best solution in $\mathcal{P}(\mathcal{G})$

A BRKGA evolves a fixed-size population, denoted as $\mathcal{P}(g) = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_t\}$, composed of t random key vectors for each generation $g = 1, 2, \dots, \mathcal{G}$. The objective function values of the individuals included into the population determine a partition: $\mathcal{P}(g) = \mathcal{P}_e(g) \cup \mathcal{P}_c(g)$, $g = 1, 2, \dots, \mathcal{G}$ (where $t = e + c$). In this regard, $\mathcal{P}_e(g) \subset \mathcal{P}(g)$ is termed *elite population* and composed of the elite individuals, whereas $\mathcal{P}_c(g) \subset \mathcal{P}(g)$ is termed *non-elite population* and contains the remaining individuals. Each random key vector, $\mathcal{P} \in \mathcal{P}(g)$, is mapped at the solution space of the optimization problem by means of a deterministic procedure termed *decoder*, denoted as $d : \mathcal{P} \rightarrow \phi$ (see Section 2.2). This way, a random key vector, $\mathcal{P}$, is decoded to a feasible solution of the optimization problem, $\phi \in \mathcal{S}_n$. Once the solution is decoded into the problem space, its fitness value, $f(\phi)$, is computed. The evolutionary dynamics of a BRKGA are as follows. At each generation g, all the elite individuals are copied from the current population $\mathcal{P}(g)$ (without any change) to the population of the next generation, $\mathcal{P}(g+1)$. Afterwards, a set $\mathcal{P}_m(g+1)$ of mutant individuals is inserted into $\mathcal{P}(g+1)$ with the goal of diversifying the search.

In this work, we propose a Hybrid Biased Random Key Genetic Algorithm (HBRKGA) approach in order to solve the QAP. Its pseudocode is depicted in Algorithm 1. It takes as parameters the number of generations, $\mathcal{G}$, the size of the population, t, the number of elite individuals, e, the number of mutant individuals, m (where $e + m \leq t$ and $2 \times e \leq t$), the crossover rate, α, and the number of facilities involved in the QAP to be solved, n. The first step of the HBRKGA is to obtain the initial population, $\mathcal{P}(1)$ (line 1) generated by a Solution Generation Procedure consisting of generating n random keys at random.

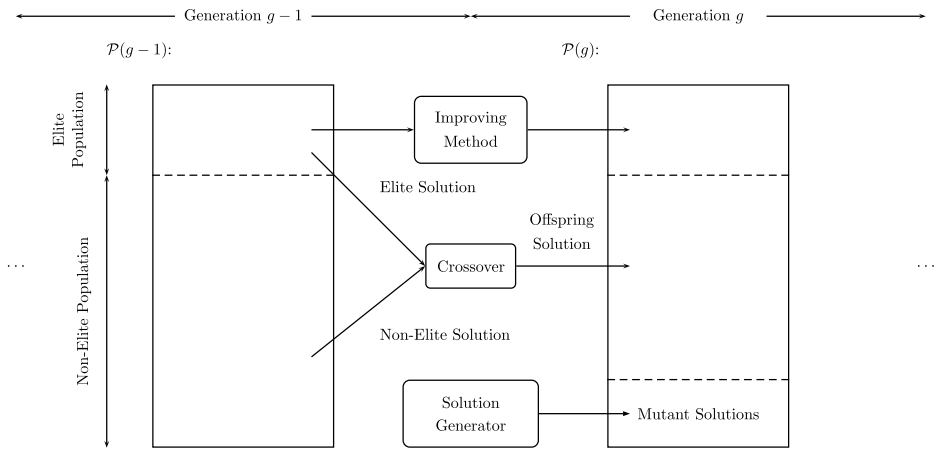

Fig. 1. Scheme of the Hybrid Biased Random Key Genetic Algorithm.

With this goal in mind, t random key vectors, composed of n random keys each one, are generated. The fitness of each individual included into the initial population $\mathcal{P}(1)$ is evaluated according to $f(\cdot)$ (line 2). The HBRKGA iterates during $\mathcal{G}$ generations (lines 3–14) in such a way that a new population is generated. The population in the generation g, $\mathcal{P}(g)$, includes the elite individuals of the previous population, that is, $\mathcal{P}_e(g-1)$ (line 4). Unlike the BRKGA, the elite individuals imported from the previous population are improved by means of an improvement method with the goal of exploiting the promising regions of the search space of the QAP found during the search (line 5). This step is later discussed in Section 2.3. Also, m mutant random key vectors are included in the new population in order to diversify the search and avoid its fast convergence (line 6). The rest of the population is filled with individuals obtained through a crossover operator applied to individuals from the previous population (lines 7–12). One of the parents involved in the crossover is selected from the elite population, $\mathcal{P}_e(g-1)$ (line 8), whereas the other one is selected from the non-elite population, that is, $\mathcal{P}_c(g-1) = \mathcal{P}(g-1) \setminus \mathcal{P}_e(g-1)$ (line 9). Section 2.4 broadly describes the crossover operator. In addition, Fig. 1 shows the flows of individuals between populations in successive generations including the inclusion of new individuals as mutants by means of the *Solution Generator* procedure (line 6). The last step of each generation evaluates the fitness of the individuals included into the current population (line 13). Finally, the HBRKGA reports the best solution included into the last population $\mathcal{P}(\mathcal{G})$ (line 15).

2.1. Coding

A feasible solution of the QAP, $\phi \in \mathcal{S}_n$, is represented as a permutation. In the random key representation, each solution is encoded by means of an array with n components, $\mathcal{P} = (p_1, p_2, \ldots, p_n)$, where n is the number of facilities (see Section 1). Each component, p_i $(i = 1, 2, \ldots, n)$, is a random key and, by definition, a real number in the interval $[0, 1)$. Fig. 2 shows an example of a random key vector of a QAP with $n = 6$ facilities.

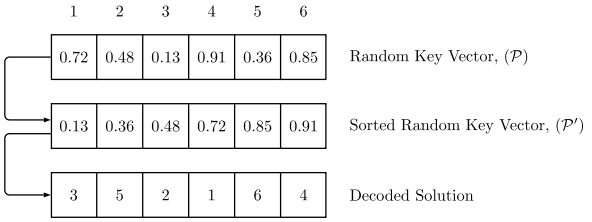

Fig. 2. Decoding process of a random key vector for the Quadratic Assignment Problem.

2.2. Decoding

The decoding process is carried out by means of a deterministic procedure termed decoder which seeks that each random key vector, $\mathcal{P} \in \mathcal{P}(g)(g = 1, 2, \ldots, \mathcal{G})$, is mapped at the solution space of the QAP. Its random keys, $p_i \in \mathcal{P}$ $(i = 1, 2, \ldots, n)$, are used to determine the assignment of facilities to the available locations. For this purpose, the random keys are sorted in an non-decreasing order, giving rise to a sorted random key vector denoted as $\mathcal{P}' = (p'_1, p'_2, \ldots, p'_n)$, where $p'_i \leq p'_{i+1} (i = 1, 2, \ldots, n-1)$. The location assigned to each facility stems from the position of the value p'_i $(i = 1, 2, \ldots, n)$ in the original random key vector, $\mathcal{P}$. Fig. 2 illustrates the decoding process of a random key vector of a QAP with $n = 6$ facilities.

2.3. Improvement method

The improvement method is aimed at exploiting the promising regions of the search space found by the HBRKGA. Specifically, its goal is to obtain a feasible solution, ϕ', for each decoded solution of the elite population, ϕ, in such a way that $f(\phi') < f(\phi)$. With this goal in mind, the *2-opt* neighbourhood structure (Burkard et al. [2]) is applied. That is, given a decoded solution, $\phi \in \mathcal{S}_n$, the *2-opt* neighbourhood, $\mathcal{N}(\phi) = \{\phi \circ (i, j) : 1 \leq i, j \leq n, i \neq j\}$, performs the transposition (i, j) by swapping the two relevant locations assigned to the facilities, i and j, respectively. In order to keep a reduced time consumption, the

Table 1
Parameter values used in the parameter setting.

Parameter	Value
t	$\in \{50, 100, 150, 200\}$
e	$\in \{5, 10, 15\}$
m	$\in \{5, 10, 15\}$
α	$\in \{0.6, 0.7, 0.8, 0.9\}$
$\mathcal{G}$	$\in \{100, 200\}$

2-opt iterates only once over each decoded solution, ϕ. The selection of the neighbour solution is performed by following the steepest descent strategy. That is, an exhaustive exploration of the neighbourhood of ϕ is carried out and that neighbour solution, $\phi' \in \mathcal{N}(\phi)$, that maximizes $\Delta = f(\phi) - f(\phi')$ is considered.

2.4. Crossover

The crossover is carried out by combining the genes of two chromosomes, rk_1 and rk_2, where $rk_1 \in \mathcal{P}_e(g-1)$ and $rk_2 \in \mathcal{P}_c(g-1)$. The combination strategy used in HBRKGA is the parameterized uniform crossover (Spears and De Jong [18]), where a crossover rate denoted as $\alpha > 0.5$ is selected. An example of this crossover strategy for the HBRKGA is as follows. Given two random keys, $rk_1 = (0.34, 0.9, 0.55, 0.71, 0.48)$ and $rk_2 = (0.64, 0.81, 0.05, 0.26, 0.17)$, and a crossover rate, $\alpha = 0.7$, the offspring of both parents for the random sequence $(0.64, 0.9, 0.56, 0.71, 0.17)$ is as follows: $rk = (0.34, 0.81, 0.55, 0.26, 0.48)$.

3. Computational results

This section is wholly devoted to demonstrate the suitable performance of the HBRKGA previously introduced in Section 2. In this regard, we perform a set of computational experiments over a representative set of instances from the most extended library of problem instances for the Quadratic Assignment Problem (QAP), the QAPLIB proposed by Burkard et al. [3]. For this representative set and in the same way as Duman et al. [9], we divided it into *sparse* and *dense* instances. Moreover, we also considered the instances proposed by Duman et al. [9]. This set of instances has been generated considering the printed circuit board problem (PCB-QAP) related to the location of electronic components. In each case, 20 executions of our HBRKGA were carried out for each problem instance. The proposed optimization technique has been implemented in Java Standard Edition 7 and executed on a computer equipped with an Intel 3.16 GHz and 4 GB of RAM.

3.1. Parameter setting

Our first goal is to conduct a proper selection of parameter values for the HBRKGA through a statistical analysis. In this regard, Table 1 shows reasonable parameter values to assess during the parameter setting. In this work, once the performance of the HBRKGA with each combination of parameter values is known, the Friedman nonparametric statistical test (Daniel [6]) is used in order to state an order of performances. In those cases in which the null hypothesis of equality of treatments is rejected, the multiple comparisons test of Friedman is used with the aim of determining the differences among combinations.

According to the aforementioned discussion, the Friedman test is applied to the average objective function value of the HBRKGA on a subset of 5 representative instances with the combinations of parameter values reported in Table 1. In this case, the Friedman test at $\alpha_{friedman} = 0.05$ significance level for the objective function values indicates that there are statistically significant differences among the combinations of parameter values. Then, in the remainder of this work, the combination $t = 100$, $e = 10$, $m = 5$, $\alpha = 0.8$, and $\mathcal{G} = 100$ is used due to the fact that it presents the best performance.

3.2. Comparison with previous approaches

In the following, we perform a comparison between the proposed HBRKGA, BRKGA, and the most competitive approximate approaches from the related literature. Namely, the Migrating Birds Optimization (MBO) recently proposed by Duman et al. [9] and the Discrete Differential Evolution Algorithm with Local Search (DDE_{LS}) proposed by Tasgetiren et al. [19]. It is worth mentioning that, the authors of the MBO only report the computational consumption of their algorithm for two instance sizes ($n = 32$ and $n = 80$). Consequently, in order to carry out a suitable comparison we estimate linearly the computational time of MBO for the remaining instance sizes by considering the two times provided in their work. As indicated by Duman et al. [9], the MBO has been executed on a computer equipped with an Intel Core2 2.83 GHz and 3 GB of RAM. On the other hand, as indicated by Tasgetiren et al. [19] the DDE_{LS} has been executed using the same termination criterion as Duman et al. [9] comparing in their paper only the quality of the solution in terms of the objective function value. Thus, for the DDE_{LS} we estimate the same computational time as MBO.

Table 2 and Table 3 show the comparison between HBRKGA, BRKGA, and MBO (Duman et al. [9]). The first column in the tables reports the properties of the instances used. That is, the identifier (*Id*) and size of the instances (n). The second column indicates the objective function value of the Best Known Solution (BKS) reported in the literature for the corresponding instance. It should be noted that those objective function values are reported without considering the results of our HBRKGA. Furthermore, the performances of MBO, BRKGA, DDE_{LS}, and HBRKGA in terms of objective value (Obj.), relative error (Gap (%)), and computational time (measured in seconds) are lastly reported.

The quality of the solutions reported by HBRKGA in Table 2 indicates that our approach, although exhibits a worse objective function value for some instances with respect to DDE_{LS}, is highly effective regardless if the instances are either sparse or dense when compared with MBO. Moreover, the results reported in Table 3, HBRKGA exhibits, on average, a better performance than MBO and DDE_{LS}. In this regard, HBRKGA is able to provide several new best solutions for the instances *B2*, *B7* and *B9*. In

Table 2
Comparison among Migrating Birds Optimization (MBO) (Duman et al. [9]), Discrete Differential Evolution Algorithm with Local Search (DDE_{LS}) proposed by Tasgetiren et al. [19], BRKGA, and HBRKGA for solving the instances from QAPLIB (Burkard et al. [3]). The horizontal line divides dense instances from sparse. Best values in bold.

Instance	BKS	MBO			DDE_{LS}		BRKGA			HBRKGA		
		Obj.	Gap (%)	t (s.)	Obj.	Gap (%)	Obj.	Gap (%)	t (s.)	Obj.	Gap (%)	t (s.)
esc32e	2	**2**	**0.00**	0.20	**2**	**0.00**	**2**	**0.00**	0.11	**2**	**0.00**	0.96
esc32f	2	**2**	**0.00**	0.20	**2**	**0.00**	**2**	**0.00**	0.14	**2**	**0.00**	0.99
esc32g	6	**6**	**0.00**	0.20	**6**	**0.00**	**6**	**0.00**	0.18	**6**	**0.00**	1.05
esc32h	438	**438**	**0.00**	0.20	**438**	**0.00**	448	2.28	0.16	**438**	**0.00**	1.11
esc64a	116	**116**	**0.00**	16.73	**116**	**0.00**	136	17.24	0.50	**116**	**0.00**	15.25
tai64c	1855928	**1855928**	**0.00**	16.73	**1855928**	**0.00**	1857646	0.09	0.50	**1855928**	**0.00**	13.78
lipa40b	476581	501792	5.29	4.33	**476581**	**0.00**	582757	22.28	0.27	**476581**	**0.00**	2.59
sko49	23420	23683	1.27	8.98	**23488**	**0.29**	24890	6.28	0.33	23586	0.71	4.89
wil50	48816	49094	0.57	9.50	**48879**	**0.13**	50352	3.15	0.40	49036	0.45	5.27
tai60b	608215054	609249020	0.17	14.67	**608519162**	**0.05**	650764660	7.00	0.56	609808716	0.26	10.80
lipa70a	169755	171130	0.81	19.83	**171096**	**0.79**	172456	1.59	0.63	171109	0.80	20.76
lipa80a	253195	255069	0.74	25.00	**254942**	**0.69**	256777	1.41	0.79	255007	0.72	33.48

Table 3
Comparison among Migrating Birds Optimization (MBO) (Duman et al. [9]), Discrete Differential Evolution Algorithm with Local Search (DDE_{LS}) proposed by Tasgetiren et al. [19], BRKGA, and HBRKGA for solving the PCB-QAP problem instances (Duman et al. [9]). Best values in bold.

Instance		BKS	MBO			DDE_{LS}		BRKGA			HBRKGA		
id	*n*		Obj.	Gap (%)	t (s.)	Obj.	Gap (%)	Obj.	Gap (%)	t (s.)	Obj.	Gap (%)	t (s.)
B1	52	1074	**1074**	**0.00**	10.53	1086	1.12	1652	53.82	0.51	1076	0.19	10.08
B2	54	764	764	0.00	11.57	760	−0.52	1170	53.14	0.44	**752**	**−1.57**	7.78
B3	52	740	762	2.97	10.53	**724**	**−2.16**	1154	55.95	6.56	746	0.81	6.56
B5	50	1462	1462	0.00	9.50	**1454**	**−0.55**	1940	32.69	0.47	1458	−0.27	5.50
B6	48	756	758	0.26	8.47	**756**	**0.00**	1154	52.65	0.39	**756**	**0.00**	4.96
B7	49	1392	1398	0.43	8.98	1396	0.29	1790	28.59	0.41	**1388**	**−0.29**	5.34
B8	47	1358	1358	0.00	7.95	**1348**	**−0.74**	1842	35.64	0.35	**1348**	**−0.74**	4.51
B9	40	718	722	0.56	4.33	716	−0.28	950	32.31	0.31	**714**	**−0.56**	2.39

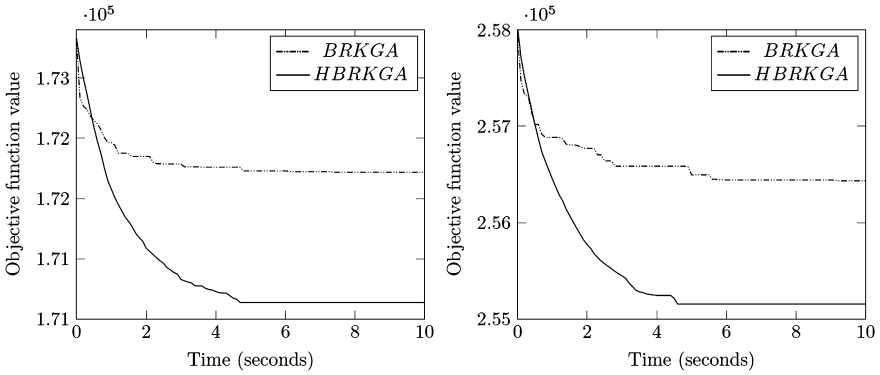

Fig. 3. Performance of BRKGA and HBRKGA for two instances selected at random.

these cases, HBRKGA reports solutions with a maximum improvement of 1.57% in the best case (instance *B2*). The implementation of the BRKGA for the QAP presented in this work exhibits a competitive behaviour in terms of computational time. However, as reported in the tables and discussed below, the quality of the solutions provided by BRKGA is not competitive in comparison with the other solution approaches.

With the aim of analyzing the overall performance of BRKGA when an improvement phase is applied over the individuals belonging to the elite population, we made a comparison between BRKGA and HBRKGA. The rationale behind this is to analyze the contribution of the improvement method to BRKGA. Fig. 3 shows the performance of both algorithms over time, measured in seconds. Two problem instances were selected at random. As can be seen, for the same amount of computational time HBRKGA exhibits better performance than BRKGA. This may indicate that the use of an improvement method within the BRKGA produces an improvement over the quality of the solutions. In this regard, at the initial phase of the search they have a quite similar behaviour, however when the search continues, HBRKGA clearly presents a better behaviour. The plots also highlight that HBRKGA quickly converges to its best solutions, this gives rise to the use of other additional stopping criterion to improve its temporal performance.

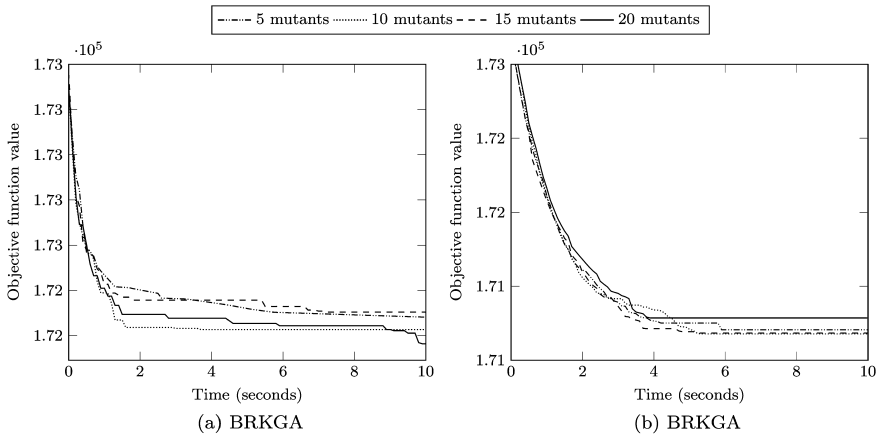

Fig. 4. Performance of BRKGA and HBRKGA for one instances from Burkard et al. [3] selected at random.

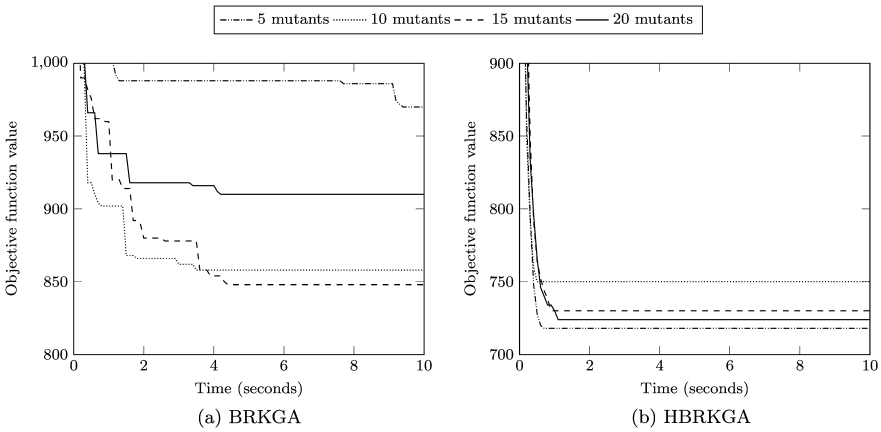

Fig. 5. Performance of BRKGA and HBRKGA for one instance from Duman et al. [9] selected at random.

In Figs. 4 and 5 we assess how BRKGA and HBRKGA respond to parameter m (number of mutant individuals in the population) along the same time limit of 10 seconds. It can be seen that for that time period and regardless of the value of m HBRKGA exhibits a better performance in terms of the evolution of the objective function value shown along the time. This indicates that improving the elite population –through the improvement method– drives to a rapid convergence and best objective values than without it. In the case of Fig. 4 within the 2 first seconds HBRKGA converges faster to better objective values than BRKGA. Similarly, as shown in Fig. 5 in less than two seconds HBRKGA is able to converge to its best objective value, where BRKGA without improvement method may require more than 10 seconds. Finally, in the case of BRKGA we can see a great variability of the objective function values at the end of the time limit.

3.3. Contribution of the improvement method in unbiased and biased random key approaches for the QAP

In order to analyze the overall performance of our approach and contextualize its behaviour, we assess the contribution that the improvement method proposed in this work has over the unbiased and biased RKGAs when is applied to the individuals belonging to the elite population. Hence, in this subsection we have made a comparison among RKGA, BRKGA, HRKGA, and HBRKGA. It should be noted that HRKGA is a RKGA in which an improvement method is applied over the elite individuals of the population.

Fig. 6 and Fig. 7 show the performance of the algorithms under analysis over time (measured in seconds). Two problem instances from each benchmark suite (QAPLIB and PCB-QAP) are selected at random. As can be seen, for the same amount of computational time, the usage of an improvement method within BRKGA and RKGA improves the quality of the solutions provided by both methods. This suggests that incorporating an improvement method within this genetic algorithms may improve their performance. Concerning HRKGA and HBRKGA, both algorithms present a homogeneous behaviour. Nevertheless, HBRKGA exhibits an slightly better behaviour than HRKGA. In the QAPLIB instances, BRKGA provides solutions with lower objective function values than RKGA. This is not reflected for the PCB-QAP instances, where both show a similar behaviour. The plots also highlight that HRKGA and HBRKGA converge quickly toward their best solutions in the case

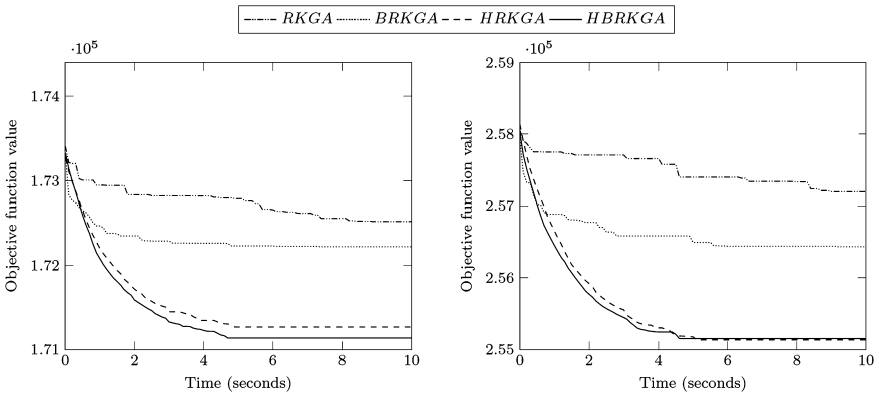

Fig. 6. Performance comparison of RKGA, BRKGA, HRKGA, and HBRKGA for two instances selected at random from the QAPLIB instances.

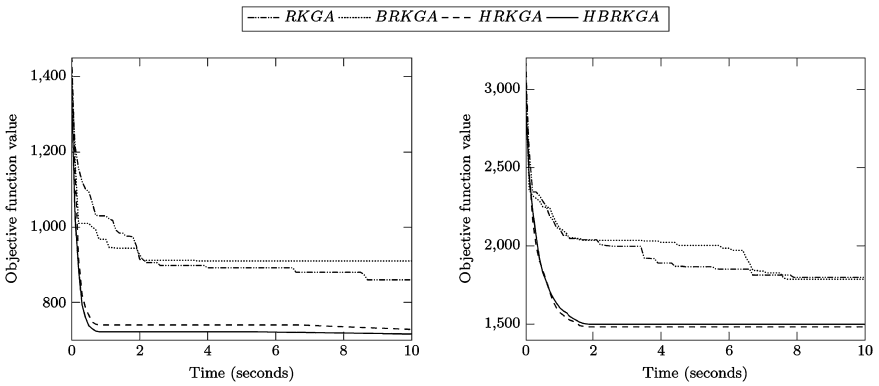

Fig. 7. Performance comparison of RKGA, BRKGA, HRKGA, and HBRKGA for two instances selected at random from PCB-QAP instances.

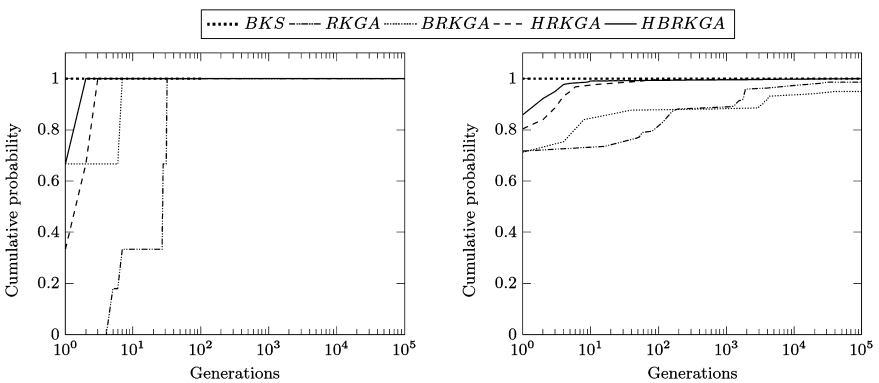

Fig. 8. Convergence comparison of RKGA, BRKGA, HRKGA, and HBRKGA for two instances selected at random from the QAPLIB instances.

of PCB-QAP. It should be pointed out that this fact makes sense with the results of HBRKGA for those instances.

Fig. 8 and Fig. 9 show the generations-to-target plots stemming from the execution of the algorithms when solving instances with well-known best objective function values. Each algorithm has been executed 10 times with a maximum of $\mathcal{G} = 10^5$ generations. As proposed by Ribeiro et al. [16], the goal is to derive empirical performance distributions of the algorithms and then estimate the probabilities that a certain algorithm requires a shorter number of generations than the remaining ones. For this experiments two problem instances from each benchmark suite (QAPLIB and PCB-QAP) have been selected at random.

In the case of the QAPLIB problem instances, in both cases the usage of a biased strategy when performing the selection of the parents exhibits a better performance either an improvement method or not is used. Moreover, HRKGA and HBRKGA converge quickly to the best solution known. In this case, the latter one presents a better behaviour than HRKGA. For the PCB-QAP instances, it can be clearly seen that, on one hand, HRKGA and HBRKGA exhibit a quite similar behaviour. On the other hand, using no im-

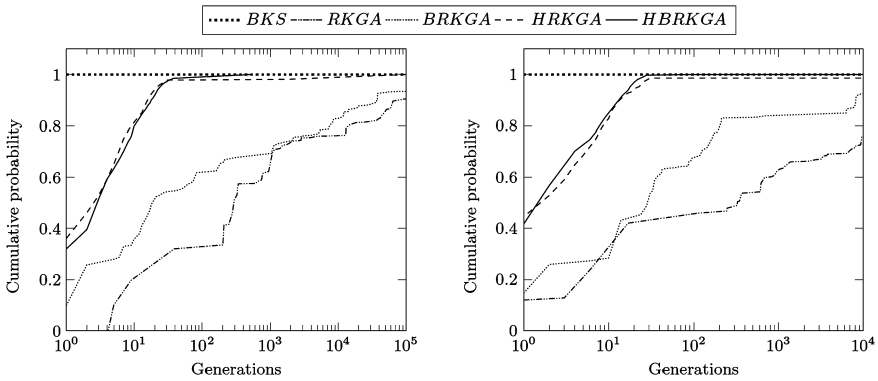

Fig. 9. Convergence comparison of RKGA, BRKGA, HRKGA, and HBRKGA for two instances selected at random from PCB-QAP instances.

provement method gives rise to a poor convergence toward the best solution. Note that for these algorithms the converge toward the best known solution requires more than 10^5 generations.

4. Conclusions and further research

In this paper, we present a Hybrid Biased Random Key Genetic Algorithm (HBRKGA) for solving the Quadratic Assignment Problem. This hybrid approach includes an improvement method over the individuals belonging to the elite sub-population. It is noticeable from the computational experiments that the proposed algorithm is able to report high-quality solutions by means of short computational times. HBRKGA provides new best solutions for some problem instances reported in the literature.

The computational results indicate that the use of an improvement method within the BRKGA framework produces an improvement of the quality of the solution. In this regard, as shown in the relevant section, within the same computational effort HBRKGA provides better solution objective function values that without its use. Moreover, HBRKGA exhibits a rapid convergence to its best solutions, which gives rise to improve its temporal behaviour by including additional stopping criterion.

Several lines are still open for further research. In the future, we are going to test the performance of HBRKGA in other related assignment problems. In this regard, we are also going to study how different configurations of the population impact on the performance of HBRKGA.

Acknowledgements

This work has been partially funded by the European Regional Development Fund, the Spanish Ministry of Economy and Competitiveness (project TIN2012-32608). Eduardo Lalla-Ruiz and Christopher Expósito-Izquierdo thank the Canary Government for the financial support they receive through their doctoral grants.

References

[1] J.C. Bean, Genetic algorithms and random keys for sequencing and optimization, ORSA J. Comput. 6 (1994) 154–160.
[2] R.E. Burkard, E. Çela, P.M. Pardalos, L.S. Pitsoulis, The quadratic assignment problem, in: D.Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Springer US, 1999, pp. 1713–1809.
[3] R.E. Burkard, S.E. Karisch, F. Rendl, Qaplib – a quadratic assignment problem library, J. Glob. Optim. 10 (4) (1997) 391–403.
[4] A.F. Chawki, O.D. Salih, An extreme point algorithm for a local minimum solution to the quadratic assignment problem, Eur. J. Oper. Res. 156 (3) (2004) 566–578.
[5] C. Chun-Hung, C.H. Sin, K. Cheuk-Lam, The use of meta-heuristics for airport gate assignment, Expert Syst. Appl. 39 (16) (2012) 12,430–12,437.
[6] W.W. Daniel, Applied Nonparametric Statistics, PWS-Kent Publishing Company, Boston, 1990.
[7] Z. Drezner, P.M. Hahn, E.D. Taillard, Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods, Ann. Oper. Res. 139 (1) (2005) 65–94.
[8] E. Duman, I. Or, The quadratic assignment problem in the context of the printed circuit board assembly process, Comput. Oper. Res. 34 (1) (2007) 163–179.
[9] E. Duman, M. Uysal, A.F. Alkaya, Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem, Inf. Sci. 217 (2012) 65–77.
[10] G. Erdoğan, B. Tansel, A branch-and-cut algorithm for quadratic assignment problems based on linearizations, Comput. Oper. Res. 34 (4) (2007) 1085–1106.
[11] M. Ericsson, M.G.C. Resende, P.M. Pardalos, A genetic algorithm for the weight setting problem in OSPF routing, J. Comb. Optim. 6 (2002) 299–333.
[12] J.F. Gonçalves, M.G.C. Resende, Biased random-key genetic algorithms for combinatorial optimization, J. Heuristics 17 (2011) 487–525.
[13] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, 1975.
[14] T.C. Koopmans, M. Beckman, Assignment problems and the location of economic activities, Econometrica 25 (1957) 53–76.
[15] E.M. Loiola, N.M. Maia de Abreu, P.O. Boaventura-Netto, P. Hahn, T. Querido, A survey for the quadratic assignment problem, Eur. J. Oper. Res. 176 (2) (2007) 657–690.
[16] C.C. Ribeiro, I. Rosseti, R. Vallejos, Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms, J. Glob. Optim. 54 (2) (2012) 405–429.
[17] S. Sahni, T. González, P-complete approximation problems, J. ACM 23 (3) (1976) 555–565.
[18] W.M. Spears, K.A.D. Jong, On the virtues of parameterized uniform crossover, in: Proceedings of the Fourth International Conference on Genetic Algorithms, 1991, pp. 230–236.
[19] M.F. Tasgetiren, Q. Pan, P. Suganthan, I. Dizbay, Metaheuristic algorithms for the quadratic assignment problem, in: 2013 IEEE Workshop on Computational Intelligence in Production and Logistics Systems, CIPLS, IEEE, 2013, pp. 131–137.
[20] X. Wu, H.D. Mittelmann, X. Wang, J. Wang, On computation of performance bounds of optimal index assignment, in: Data Compression Conference, 2010, pp. 189–198.