SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
VOLUME 5 NO 4, 2017
ISSN 2054-7390

Random Key Cuckoo Search for the Quadratic Assignment Problem

Aziz Ouaarab*‡, Bela¨ıd Ahiod*, Xin-She Yang†
*LRIT, Associated Unit to the CNRST(URAC) no 29, Mohammed V-Agdal University, Morocco
†School of Science and Technology, Middlesex University, The burroughs, London NW4 4BT, UK
‡School of Technology Essaouira, Cadi Ayyad University. Morocco
aziz.ouaarab@gmail.com; Ahiod@fsr.ac.ma; x.yang@mdx.ac.uk

ABSTRACT

This paper proposes an adaptation of the Random- Key Cuckoo Search (RKCS) algorithm for solving the famous Quadratic Assignment Problem (QAP). We used a simplified and efficient random-key encoding scheme to convert a continous space (real numbers) into a combinatorial space. We also considered the displacement of a solution in both spaces by using Le´vy flights. The performance of the RKCS for QAP is tested against a set of benchmarks of QAP from the well-known QAPLIB library, and the comparison with a set of other methaheuristics is also carried out.

Index Terms—Nature-Inspired Metaheuristic, Cuckoo Search, Le´vy Flights, Combinatorial Optimization, Quadratic Assign- ment Problem, Random-Key.

1 Introduction

NP-hard problems [17] are very challenging to solve. It is also the most complicated among combinatorial optimization problems. The main difficult of such problems is that the number of combinations grows exponentially with the problem size. Quadratic assignment problem [7] is one of the problems that belongs to this class.

Quadratic Assignment Problem (QAP) is a combinatorial optimization problem that is applied to solve various problems in many fields such as Steinberg Wiring Problem [5], Hospital Layout [13], Dartboard Design [12], and many other applica- tions [7], [14]. Problems such as QAP do not have an efficient algorithm to solve them exactly. It is practically very difficult to get a solution of optimal quality and in a reduced runtime simultaneously. This requires some heuristic algorithms that can find good (not necessarily optimal) solutions in a good runtime by trial and error. Approximate algorithms such as metaheuristics [4] are actually the best choice to solve many combinatorial optimization problems. They are characterized by their simplicity and flexibility while demonstrating remark- able effectiveness. Many metaheuristics are proposed to solve QAP. Existing studies include Genetic Algorithms (GA) [1] that used a sequential constructive crossover, a modified Particle Swarm Optimisation (PSO) [21], Ant Colony Optimisation (ACO) [11], and many other examples [22], [15].

Among the most difficult issues that arises when solving a combinatorial optimization problem with a metaheuristic, is how to move in the combinatorial solution space without affecting the performance of

DOI: 10.14738/tmlai.54.3666
Publication Date: 28th August, 2017
URL: http://dx.doi.org/10.14738/tmlai.54.3666

the metaheuristic. Several meta- heuristics are designed in principle for continuous optimization problems. So, the question is how to treat combinatorial problems properly without losing the good performance of these metaheuristics. In this paper, we used the Random-Key Cuckoo Search (RKCS) algorithm by using the random key encoding scheme to represent a position, found by the cuckoo search algorithm, in the QAP solution space.

This work presents a novel approach using the improved cuckoo search algorithm [26], based on random keys [3], with a simple local search procedure to solve QAP.

The rest of this paper is organized as follows: Section 2 introduces briefly the QAP. Section 3, first, introduces the standard cuckoo search and its improved version. Section 4 presents the random-key encoding scheme, while Section 5 describes how CS solves QAP by using Random key. Then, Section 6 presents results of numerical experiments on a set of QAP benchmarks from the QAPLIB library [8]. Finally, Section 7 concludes with some discussions and future directions.

2 Quadratic Assignment Problem

The Quadratic Assignment Problem is a combinatorial optimisation problem, which tries to minimize the total cost of building and operating the facilities knowing that the benefit resulting from an economic activity at any site is depending on the sites of the other facilities. The solution space in QAP is considered as a set of all potential assignments of the facilities to the possible sites.

A selected solution S is the permutation φ of a given set Q = {1, 2, . . . , N } where N is the instance dimension, it is also the number of sites and facilities, φ(i) = k means that the facility i is assigned to the site k.

The objective problem is to find a permutation φ = (φ(1), φ(2), ..., φ(N)) that minimizes

$$\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij} b_{\phi(i)\phi(j)} \qquad (1)$$

where a is the flow matrix, and aij is the flow between facilities i and j, and b denotes the distance matrix. So, the distance from facility i to j takes the value of bφ(i)φ(j).

Here, φ(i) is the location assigned to facility i. The aim is to minimize the sum of products flow×distance [30].

3 Cuckoo Search Algorithm

In the aim to increase their survival chances and reduce the probability of abandoning eggs by the host birds, cuckoos adopt many strategies and tricks. These strategies are mimicked successfully and designed in the well known Cuckoo Search (CS) algorithm [34]. Cuckoo search introduces Le´vy flights [28] for generating a new good solution. Le´vy flights, named after the French mathematician Paul Le´vy, represent a model of random walks characterized by their step lengths which obey a power-law distribution [6]. CS is widely applied to solve many combinatorial optimization problems such as Travelling Salesman Problem [25], [27], Flow Shop Schedul- ing Problem [20], Knapsack Problem [19].

The first version of CS, which is developed by Xin-She Yang and Suash Deb, is summarized as the following ideal rules: (1) Each cuckoo lays one egg at a time and selects a nest randomly; (2) The best nest with the

highest quality of egg can pass onto the new generations; (3) The number of host nests is fixed, and the egg laid by a cuckoo can be discovered by the host bird with a probability pa $\in$ [0, 1]. pa is also a switch parameter to control the balanced combination of local

explorative random walk and the global explorative random walk. The local random walk can be written as :

$$x_i^{t+1} = x_i^t + \alpha s \otimes H(p_a - \epsilon) \otimes (x_j^t - x_k^t), \qquad (2)$$

where x_j^t and x_k^t are two different solutions selected randomly by random permutation, H(u) is a Heaviside function, ϵ is a random number drawn from a uniform distribution, and s is the step size. On the other hand, the global random walk is carried out by using Lévy flights

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda), \qquad (3)$$

Where

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \qquad (4)$$

Here $\alpha > 0$ is the step size scaling factor, which should be related to the scales of the problem of interest. Lévy flights have an infinite variance with an infinite mean [34]. Here s_0 is a constant, which can be take as 0.01 to 0.1.

Before applying CS to solve QAP, as described in the Algorithm 1, we will consider an improved version of CS [26]. This improvement introduces a new category p_c of cuckoos that can engage a kind of surveillance on nests likely to be a host. So, around the pertinent solutions, this portion p_c intensify the search process to find a new better solution via Lévy flights.

Algorithm 1 Improved CS Algorithm

1: Objective function $f(x), x = (x_1, \ldots, x_N)^T$
2: Generate initial population of n host nests $x_i (i = 1, \ldots, N)$
3: **while** ($t <$MaxGen) or (stop criterion) **do**
4: **Start searching with a fraction** (p_c) **of smart cuckoos**
5: Get a cuckoo randomly by Lévy flights
6: Evaluate its quality/fitness F_i
7: Choose a nest among n (say, j) randomly
8: **if** $(F_i > F_j)$ **then**
9: replace j by the new solution;
10: **end if**
11: A fraction (p_a) of worse nests are abandoned and new ones are built;
12: Keep the best solutions (or nests with quality solutions);
13: Rank the solutions and find the current best
14: **end while**
15: Postprocess results and visualization

4 Random-Key Encoding Scheme

Random-key encoding scheme [3], [27] is an interesting procedure that can be useful to pass from a continuous space to the combinatorial space. In general, the position in the continuous space is represented by a vector of real numbers. To have a projection in the combinatorial space, random-key (RK) associates each real number with a weight. These weights are used to generate one combination as a solution. The random real numbers drawn uniformly from $[0, 1)$ compose a vector showed in Figure 1. On the other hand, the combinatorial vector is composed of integers ordered according to the weights of real numbers in the first vector, illustrated in the Figure 1.

Random keys:	0.8	0.5	0.7	0.1	0.4	0.2
Decoded as:	6	4	5	1	3	2

Figure. 1. Random key encoding scheme

5 Random Key Cs For Qap

A QAP solution is a vector of N integers. Each integer is the facility index and its order in the vector is the corresponding site index. By considering Figure 1, we can say that resulting vector in this figure is a QAP solution. So to move from the current solution to a new one we can just perturb the first vector that contains the real numbers. This perturbation is performed via Lévy flights. In the case of big jumps we per- form a random pairwise interchanges of vector integers. It is a series of chained swaps $\phi_{\xi(1)\xi(2)}$, $\phi_{\xi(2)\xi(3)}, \ldots, \phi_{\xi(\rho-1)\xi(\rho)}$, where $\xi(1), \xi(2), \ldots, \xi(\rho)$ is a sequence of random integers between 1 and N such that $\xi(i) = \xi(\mathbf{j}), i, \mathbf{j} = 1, 2, \ldots, \rho, i = j, 1 < \rho \leq N$. The mutated solution π can thus be represented as a composition $((((\pi \oplus \phi_{\xi(1)\xi(2)}) \oplus \phi_{\xi(2)\xi(3)})\oplus, \ldots) \oplus {}_{\xi(i)\xi(i+1)}) \oplus \ldots) \oplus \phi_{\xi(\rho-1)\xi(\rho)}$, where π is the current solution. This series of chained swaps is called the controlled chained mutation (CCM) [23]. The parameter ρ is the mutation rate. This allows an improved way to balance the search for solutions in local areas as well as global areas.

To detect the best solution in the found area, a simple local search (Steepest Descent [2]) is performed. In this local search method we used 'swap' move as described in Figure 2 (In this example we chose to swap the facilities of the sites 2 and 6, which are 1 and 9.) which moves from a placement φ to a neighbour placement π by applying a swap between facilities r and s:

$$\pi(k) = \phi(k), \quad \forall k \neq r, s \tag{5}$$

$$\pi(r) = \phi(s), \quad \pi(s) = \phi(r) \tag{6}$$

To gain some precious time, estimations are generally performed. They restrict positively the choice of passing to a new good solution.

In the case of symmetrical matrices with a null diagonal the cost $\delta(\phi, r, s)$ of a move is given by:

$$\delta(\phi, r, s) = \sum_{i=1}^{N}\sum_{j=1}^{N}(a_{ij}b_{\phi(i)\phi(j)} - a_{ij}b_{\pi(i)\pi(j)}) \quad (7)$$

$$= 2.\sum_{k\neq r,s}(a_{sk} - a_{rk})(b_{\phi(s)\phi(k)} - b_{\phi(r)\phi(k)}) \quad (8)$$

In the minimization case, the swap move is done only if the new solution cost is less than the current one. Obviously, this process is repeated until no further improvement is possible or when a given number of steps is reached.

Steepest Descent is a simple local search method that can be easily trapped in a local minimum and, generally, it cannot find good quality solutions. We choose this simplified local search "Steepest Descent" method to show the performance of CS combined with RK for QAP. It allowed us to generate solutions of good quality, without introducing an advanced local search method.

6 Experimental Results

We will show some results of running RKCS to solve a set of bechmark instances [29], [31], [32], [10], [9], [13], [18], [33], [24] of QAP from the QAPLIB library [8]. Forty-six instances are considered with sizes ranging from 12 to 100 facilities. The numerical value in the name of an instance represents the number of provided facilities, e.g., the instance named sko90 has 90 facilities. We note that for each instance, 10 independent runs are carried out. These results describe the performance of this first version approach.

In Table II, we compare RKCS with two algorithms based on Genetic Algorithm [1] (SCX) and Ant Colonies [15] (HAS- QAP). Another comparisons are carried out to a Particle Swarm Optimization based algorithm [16] (HPSO) in Ta- ble III. We have implemented RKCS algorithm using Java under 32 bit MS Windows Seven operating system. Experi- ments are conducted on a laptop with Intel(R) Core™ 2 Duo 2.00 GHz CPU, and 3 GB of RAM. SCX algorithm has been encoded in Visual C++ and run on a PC with Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and 8.00GB RAM under MS Windows Seven. For HAS-QAP it was not possible to obtain the hardware configurations used in the experiments. HPSO has been run on Intel pentium core 2 Duo Q9950 (2.83GHz).

TABLE I
PARAMETER SETTINGS FOR RKCS ALGORITHM.

Parameter	Value	Meaning
n	20	Population size
p_c	0.6	Portion of smart cuckoos
p_a	0.2	Portion of bad solutions
$MaxGen$	500	Maximum number of iterations
α	0.01	Step size
λ	1	Index
ρ	$Levy$	Mutation rate

The properly selected parameter values used for the experiments of RKCS algorithm are shown in Table I. These values are selected, based on some preliminary trials, and gave the best results concerning both the solution quality and the computational time. In each case study, 10 independent runs of RKCS with these parameters are carried out.

TABLE II
COMPARING RKCS WITH GENETIC ALGORITHM (SCX) [1] AND ANT COLONIES (HAS-QAP) [15].

		SCX		HAS-QAP		RKCS	
Instance	Bkv	PDbest(%)	time(s)	PDbest(%)	time(s)	PDbest(%)	time(s)
tai20a	703482	4.50	6	1.483	3	**0.0**	20
tai20b	122455319	6.56	6	0.243	3	**0.0**	0.5
tai25a	1167256	4.58	9	2.527	5	**0.0**	39
tai25b	344355646	5.24	9	0.133	5	**0.0**	1
tai30a	1818146	4.29	13	2.600	9	**0.0**	74
tai30b	637117113	8.78	13	0.260	9	**0.0**	5
tai35a	2422002	4.95	18	2.969	15	0.92	96
tai35b	283315445	5.00	18	0.343	15	**0.0**	11
tai40a	3139370	4.53	24	3.063	24	0.79	141
tai40b	637250948	7.30	24	0.280	24	**0.0**	11
tai50a	4938796	4.51	37	3.487	50	1.34	258
tai50b	458821517	5.60	37	0.291	50	**0.0**	163
tai60a	7205962	4.54	54	3.686	88	1.27	433
tai60b	608215054	5.12	54	0.313	90	**0.0**	133
tai80a	13499184	4.35	95	2.996	220	1.53	1067
tai80b	818415043	6.63	95	1.108	225	**0.0**	1033
sko42	15812	-	-	0.654	25	**0.0**	87.06
sko49	23386	-	-	0.661	45	0.05	245.17
sko56	34458	-	-	0.729	69	0.02	356.25
sko64	48498	-	-	0.504	105	0.00	521.09
sko72	66256	-	-	0.702	153	0.11	820.53
sko81	90998	-	-	0.493	222	0.09	1090.11
sko90	115534	-	-	0.591	307	0.04	1469.63

In the following three tables of experimental results, the first column 'Instance' shows the name of the instance, while the column 'Bkv' shows the optimal solution value taken from the QAPLIB [8]. In Table II, the column 'PDbest(%)' gives the percentage deviation of the best solution value over the optimal solution value of 10 runs, and the column 'time' denotes the average run time. In Table III, the remaining columns 'mean' and 'Stdev' are the mean and standard deviation of solution cost (expressed as Average Percentage Deviation (APD), which is the percentage by which the cost exceeds the Bkv). The last Table IV which gives statistic results of testing RKCS on some QAP instances contains nine columns. The column 'best' shows the value of the best found solution. The column 'average' gives the average solution value of the 10 independent runs, the column 'worst' shows the value of the worst solution found by RKCS. The column 'SD' denotes the standard deviation. The column 'PDav (%)' denotes the percentage deviation of the average solution value over the optimal solution value of 10 runs. Bold values indicate that the solutions found have the same length as the Bkv. The percentage deviation of a solution to the best known solution (or optimal solution if known) is given by the Equation 9.

$$PDsolution(\%) = \frac{solution\ value - best\ known\ solution\ value}{best\ known\ solution\ value} \times 100 \quad (9)$$

The three tables show that RKCS can be a useful tool to switch from continuous search space to combinatorial one. It also facilitates a better control of the balance between intensification and diversification through local and global random walks.

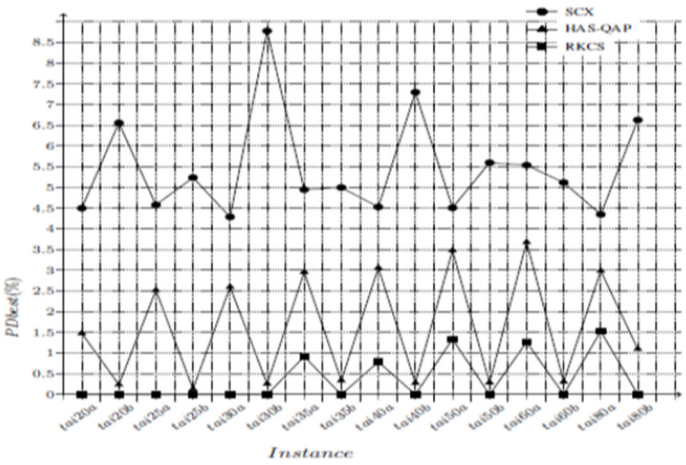

Fig. 3. PDbest(%) for 16 QAPLIB instances.

SCX performs a modified sequential constructive crossover operators in genetic algorithms, without using any robust local search technique to improve the solution quality. HAS-QAP is a hybrid ant colony system coupled with a simple local search that consists in applying a complete neighborhood examination twice with first improving strategy [15]. These two methods are compared with RKCS in Table II and Figure 3. We can observe that the difference between RKCS and the two methods is significant. This can be explained basically by a good balance between intensification and diversification, an intelligent use of Lévy flights and the reduced number of parameters.

Table III, shows the comparison of RKCS and HPSO. In this hierarchical particle swarm optimization, particles are arranged in a tree-like hierarchy, where the better-performing particles float towards the top of the hierarchy, and each particle is influenced by its immediate parent in the tree [16]. We have used in this comparison, the mean and the standard deviation. The results prove that RKCS outperforms HPSO in solving the tested QAP instanced. One of the advantages of the RKCS is the relatively independent of cuckoos in the search process for the best solution, and the use of several search strategies adopted by each category of cuckoos, without using any advanced local search method.

These results show that RKCS can be adapted easily to solve QAP. Therefore, we can say that the random-key en- coding scheme can be a very useful tool for switching from continuous to combinatorial spaces. It allows operators of the continuous space to behave freely, then projecting the changes made by these operators in the combinatorial space. It also facilitates a better control in balancing intensification and diversification through Lévy flights, which make intensified small steps in a limited region followed by a big explorative jump to a distant region. Using the real numbers, Lévy flights can easily act with the notion of distance, and can define clearly small or big steps. Then RK projects these changes in the space of QAP solutions.

TABLE III
COMPARISON OF RKCS WITH HPSO A PARTICLE SWARM OPTIMISATION BASED ALGORITHM [16].

		HPSO		*RKCS*				*HPSO*		*RKCS*	
Instance	Bkv	mean	Stdev	mean	Stdev	Instance	Bkv	mean	Stdev	mean	Stdev
bur26a	5426670	1.9	0.22	**0.0**	**0.0**	tai12a	224416	8.11	1.42	**0.0**	**0.0**
bur26b	3817852	1.96	0.41	**0.0**	**0.0**	tai12b	39464925	4.82	1.79	**0.0**	**0.0**
bur26c	5426795	2.3	0.24	**0.0**	**0.0**	tai15a	388214	7.28	0.89	0.02	0.00
bur26d	3821225	2.18	0.42	**0.0**	**0.0**	tai15b	51765268	1.05	0.15	**0.0**	**0.0**
bur26e	5386879	2.14	0.42	**0.0**	**0.0**	tai20a	703482	12.04	0.73	0.4	0.36
bur26f	3782044	2.39	0.59	**0.0**	**0.0**	tai20b	122455319	9.67	1.04	**0.0**	**0.0**
bur26g	10117172	2.07	0.22	**0.0**	**0.0**	tai25a	1167256	11.62	0.45	0.81	0.62
bur26h	7098658	2.27	0.4	**0.0**	**0.0**	tai25b	344355646	27.12	4.34	**0.0**	**0.0**
chr25a	3796	168.41	11.79	0.25	0.25	tai30a	1818146	12.25	0.41	0.89	0.35
els19	17212548	27.04	4.91	**0.0**	**0.0**	tai30b	637117113	25.18	3.28	**0.0**	**0.0**
kra30a	88900	26.85	1.09	0.16	0.16	tai35a	2422002	13.19	0.38	1.24	0.30
kra30b	91420	24.98	1.23	0.02	0.02	tai35b	283315445	27.86	2.42	**0.0**	**0.0**
wil50	48816	8.69	0.26	0.04	0.04	tai40a	3139370	13.52	0.32	1.23	0.37
nug30	6124	16.85	0.79	0.04	0.04	tai40b	637250948	35.94	2.3	**0.0**	**0.0**
nug20	2570	11.65	0.95	**0.0**	**0.0**	tai50a	4938796	13.87	0.27	1.63	0.22
sko42	15812	16.54	0.46	0.03	0.03	tai50b	458821517	34.88	1.9	0.00	0.00
sko49	23386	15.52	0.42	0.15	0.07	tai60a	7205962	13.54	0.3	1.69	0.38
sko56	34458	15.88	0.43	0.15	0.12	tai60b	608215054	36.5	1.28	**0.0**	**0.0**
sko64	48498	14.68	0.42	0.14	0.12	tai80a	13499184	12.46	0.18	1.81	0.26
sko72	66256	14.52	0.26	0.24	0.11	tai80b	818415043	34.48	0.94	0.07	0.06
sko81	90998	13.98	0.28	0.23	0.13	tai100a	21052466	11.68	0.14	1.58	0.25
sko90	115534	13.79	0.23	0.25	0.16	tai100b	1185996137	33.03	0.96	0.23	0.13

TABLE IV
RESULTS OF APPLYING RKCS ON A SET OF QAP INSTANCES.

Instance	Bkv	Best	Average	Worst	SD	PDav(%)	PDbest(%)	time(s)
bur26a	5426670	**5426670**	5426670.00	5426670	0.0	0.0	0.0	0.55
bur26b	3817852	**3817852**	3817852.00	3817852	0.0	0.0	0.0	0.76
bur26c	5426795	**5426795**	5426795.00	5426795	0.0	0.0	0.0	1.16
bur26d	3821225	**3821225**	3821225.00	3821225	0.0	0.0	0.0	1.77
bur26e	5386879	**5386879**	5386879.00	5386879	0.0	0.0	0.0	1.18
bur26f	3782044	**3782044**	3782044.00	3782044	0.0	0.0	0.0	0.60
bur26g	10117172	**10117172**	10117172.00	10117172	0.0	0.0	0.0	1.73
bur26h	7098658	**7098658**	7098658.00	7098658	0.0	0.0	0.0	0.75
chr25a	3796	**3796**	3805.75	3874	9.75	0.25	0.0	13.76
els19	17212548	**17212548**	17212548.00	17212548	0.0	0.0	0.0	0.22
kra30a	88900	**88900**	89048.75	90090	148.75	0.16	0.0	9.56
kra30b	91420	**91420**	91446.25	91490	26.25	0.02	0.0	30.05
wil50	48816	**48816**	48839.75	48886	23.75	0.04	0.0	220.50
nug30	6124	**6124**	6126.50	6128	2.5	0.04	0.0	39.90
nug20	2570	**2570**	2570.00	2570	0.0	0.0	0.0	0.67
tai12a	224416	**224416**	224416	224416	0.0	0.0	0.0	0.02
tai12b	39464925	**39464925**	39464925	39464925	0.0	0.0	0.0	0.03
tai15a	388214	**388214**	388224.8	388250	12.97	0.02	0.0	4.16
tai15b	51765268	**51765268**	51765268	51765268	0.0	0.0	0.0	0.06
tai20a	703482	**703482**	706342.0	708654	2565.75	0.40	0.0	19.94
tai20b	122455319	**122455319**	122455319	122455319	0.0	0.0	0.0	0.44
tai25a	1167256	**1167256**	1176732.8	1181894	7316.99	0.81	0.0	39.45
tai25b	344355646	**344355646**	344355646	344355646	0.0	0.0	0.0	1.13
tai30a	1818146	**1827982**	1834456.2	1843032	6474.19	0.89	0.54	74.01
tai30b	637117113	**637117113**	637117113	637117113	0.0	0.0	0.0	4.93
tai35a	2422002	2444344	2452223.4	2460532	7270.30	1.24	0.9	95.63
tai35b	283315445	**283315445**	283315445	283315445	0.0	0.0	0.0	10.52
tai40a	3139370	3164372	3178239.4	3192798	11917.91	1.23	0.79	140.86
tai40b	637250948	**637250948**	637250948	637250948	0.0	0.0	0.0	10.69
tai50a	4938796	5005070	5019579.6	5027586	11282.49	1.63	1.34	258.518
tai50b	458821517	**458821517**	458842516.4	458969332	20999.39	0.00	0.0	163.532
tai60a	7205962	7297618	7328396.4	7359568	27787.29	1.69	1.27	433.493
tai60b	608215054	**608215054**	608215054.0	608215054	0.0	0.0	0.0	133.697
tai64c	1855928	**1855928**	1855928.0	1855928	0.0	0.0	0.0	115.06
tai80a	13499184	13705886	13744806.8	13781908	35650.82	1.81	1.53	1067.45
tai80b	818415043	**818415043**	818994242.5	823187691	571643.11	0.07	0.0	1033.37
tai100a	21052466	21322258	21385568.0	21421678	53170.40	1.58	1.28	1993.95
tai100b	1185996137	1187179912	1188774344.9	1189986795	1561365.85	0.23	0.09	2221.83
sko42	15812	**15812**	15816.7	15844	4.75	0.03	0.0	87.06
sko49	23386	23398	23421.5	23450	17.80	0.15	0.05	245.17
sko56	34458	34466	34511.0	34584	42.16	0.15	0.02	356.25
sko64	48498	48502	48570.5	48686	60.57	0.14	0.00	521.09
sko72	66256	66332	66420.0	66490	79.25	0.24	0.11	820.53
sko81	90998	91088	91208.7	91380	120.75	0.23	0.09	1090.11
sko90	115534	115582	115832.2	116042	188.99	0.25	0.04	1469.63
sko100a	152002	152324	152470.2	152692	121.95	0.30	0.21	2287.07

7 Conclusion

Random Key Cuckoo Search (RKCS) is designed to be easily adapted to solve many combinatorial optimization problems such as the Travelling Salesman Problem and the Quadratic Assignment Problem. Indeed, we have shown that it is easy to move in the continuous space by using Le´vy flights and project these moves in appropriate combinatorial space, of the treated problem, via random key. It proved a good balance between intensification and diversification through Le´vy flights. In this paper, we proposed an application of RKCS to solve QAP. During the design of RKCS, we have focused on the possibility of facilitating the reuse of this metaheuristic in a direct adaptation to solve various combinatorial optimization problems.

The results are promising, but there is still room for improvement. For example, for the moment, the random key is uniform, and it may be useful to investigate if how any non- uniform decoding of random keys may affect the performance.

In addition, it will be also useful to extend the proposed approach to study larger scale benchmarks. Furthermore, the proposed RKCS may also be useful to solve other combina- torial optimization problems.

REFERENCES

[1] Ahmed, Z.: A simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem. Journal of Scientific and Industrial Research (2014)

[2] Avriel, M.: Nonlinear programming: analysis and methods. Courier Dover Publications (2012)

[3] Bean, J.: Genetic algorithms and random keys for sequencing and optimization. ORSA journal on computing 6, 154–154 (1994)

[4] Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR) 35(3), 268–308 (2003)

[5] Brixius, N.W., Anstreicher, K.M.: The Steinberg wiring problem. SIAM (2001)

[6] Brown, C.T., Liebovitch, L.S., Glendon, R.: Le´vy flights in dobe ju/?hoansi foraging patterns. Human Ecology 35(1), 129–138 (2007)

[7] Burkard, R.E.: Quadratic assignment problems. Springer (2013)

[8] Burkard, R.E., Karisch, S.E., Rendl, F.: Qap–a quadratic assignment problem library. Journal of Global Optimization 10(4), 391–403 (1997)

[9] Burkard, R.E., Offermann, D.M.J.: Entwurf von schreibmaschinen- tastaturen mittels quadratischer zuordnungsprobleme. Zeitschrift fu¨ r Operations Research 21(4), B121–B132 (1977)

[10] Christofides, N., Benavent, E.: An exact algorithm for the quadratic assignment problem on a tree. Operations Research 37(5), 760–768 (1989)

[11] Demirel, N.C¸ ., Toksarı, M.D.: Optimization of the quadratic assignment problem using an ant colony algorithm. Applied Mathematics and Computation 183(1), 427–435 (2006)

[12] Eiselt, H.A., Laporte, G.: A combinatorial optimization problem arising in dartboard design. Journal of the Operational Research Society pp. 113–118 (1991)

[13] Elshafei, A.N.: Hospital layout as a quadratic assignment problem. Operational Research Quarterly pp. 167–179 (1977)

[14] FlNKE, G., Burkard, R., Rendl, F.: Quadratic assignment problems. Surveys in combinatorial optimization p. 61 (2011)

[15] Gambardella, L.M., Taillard, E., Dorigo, M.: Ant colonies for the quadratic assignment problem. Journal of the operational research society pp. 167–176 (1999)

[16] Helal, A.M., Abdelbar, A.M.: Incorporating domain-specific heuristics in a particle swarm optimization approach to the quadratic assignment problem. Memetic Computing 6(4), 241–254 (2014)

[17] Hochbaum, D.S.: Approximation algorithms for NP-hard problems. PWS Publishing Co. (1996)

[18] Krarup, J., Pruzan, P.M.: Computer-aided layout design. In: Mathemat- ical programming in use, pp. 75–94. Springer (1978)

[19] Layeb, A.: A novel quantum inspired cuckoo search for knapsack problems. International Journal of Bio-Inspired Computation 3(5), 297–305 (2011)

[20] Li, X., Yin, M.: A hybrid cuckoo search via le´vy flights for the permutation flow shop scheduling problem. International Journal of Production Research 51(16), 4732–4754 (2013)

[21] Mamaghani, A.S., Meybodi, M.R.: Solving the quadratic assignment problem with the modified hybrid pso algorithm. In: Application of Information and Communication Technologies (AICT), 2012 6th International Conference on, pp. 1–6. IEEE (2012)

[22] Maniezzo, V., Colorni, A.: The ant system applied to the quadratic as- signment problem. Knowledge and Data Engineering, IEEE Transactions on 11(5), 769–778 (1999)

[23] Misevicius, A.: Restart-based genetic algorithm for the quadratic assign- ment problem. In: Research and Development in Intelligent Systems XXV, pp. 91–104. Springer (2009)

[24] Nugent, C.E., Vollmann, T.E., Ruml, J.: An experimental comparison of techniques for the assignment of facilities to locations. Operations research 16(1), 150–173 (1968)

[25] Ouaarab, A., Ahiod, B., Yang, X.S.: Discrete cuckoo search algorithm for the travelling salesman problem. Neural Computing and Appli- cations pp. 1–11 (2013). DOI 10.1007/s00521-013-1402-2. URL http://dx.doi.org/10.1007/s00521-013-1402-2

[26] Ouaarab, A., Ahiod, B., Yang, X.S.: Improved and discrete cuckoo search for solving the travelling salesman problem. In: Cuckoo Search and Firefly Algorithm, pp. 63–84. Springer (2014)

[27] Ouaarab, A., Ahiod, B., Yang, X.S.: Random-key cuckoo search for the travelling salesman problem. Soft Computing pp. 1–8 (2014). DOI 10.1007/s00500-014-1322-9. URL http://dx.doi.org/10.1007/s00500-014-1322-9

[28] Shlesinger, M.F., Zaslavsky, G.M., Frisch, U.: Le´vy flights and related topics in physics. In: Levy flights and related topics in Physics, vol. 450 (1995)

[29] Skorin-Kapov, J.: Tabu search applied to the quadratic assignment problem. ORSA Journal on computing 2(1), 33–45 (1990)

[30] Taillard, E.: Robust taboo search for the quadratic assignment problem. Parallel computing 17(4), 443–455 (1991)

[31] Taillard, E.: Robust taboo search for the quadratic assignment problem. Parallel computing 17(4), 443–455 (1991)

[32] Taillard, E.D.: Comparison of iterative searches for the quadratic assign- ment problem. Location science 3(2), 87–105 (1995)

[33] Wilhelm, M.R., Ward, T.L.: Solving quadratic assignment problems by simulated annealing. IIE transactions 19(1), 107–119 (1987)

[34] Yang, X.S., Deb, S.: Cuckoo search via le´vy flights. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, pp. 210–214. IEEE (2009)

