Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 32 (2014) 29 – 36

Procedia
Computer Science

5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)

A New Recombination Operator for the Genetic Algorithm Solution of the Quadratic Assignment Problem

Umut Tosun*

Baskent University Department of Computer Engineering, Engineering Faculty Baglica Campus, Ankara 06530, Turkey

Abstract

The Quadratic Assignment Problem (*QAP*) is a well known combinatorial optimization problem with a diverse set of applications. It can be transformed into many problems such as the travelling salesman, weapon target assignment, and query optimization in distributed databases. Exhaustive search methods are inadequate to solve large data sets. Genetic algorithms and tabu search meta-heuristics may provide near optimal solutions for large *QAP* instances taking a reasonable time to complete. In this paper, we present a new recombination operator based on *Order-1* crossover algorithm. The suggested approach runs quick sort partitioning algorithm to generate different chromosomes from partitions. The minimum cost partition produces offsprings with the other chromosome. The proposed approach shows outstanding performance especially for instance sizes smaller than 50 with respect to the optimal results proposed in *QAPLIB*.

Keywords: Genetic Algorithm; Order-1 Crossover; Optimization; Quadratic Assignment Problem

1. Introduction

A fundamental class of optimization problems involves assigning assets to tasks to minimize a desired cost function. These problems are categorized as Assignment Problems[1]. Variations of these problems have been studied over the years with a wide range of applications in the domains of telecommunications, transportation systems and signal processing[2]. The classical approach to the Quadratic Assignment Problem (*QAP*) was first introduced by Koopmans and Beckmann[3] as a mathematical model for the location of indivisible economic activities. Since then it has been one of the most interesting challenges for scientists having been used for modelling a great variety of problems. Typewriter keyboard design, backboard wiring[4], layout design[5], turbine balancing[6], scheduling[7], and data allocation[8] are some of the problems that have been successfully modeled as *QAP*. The service allocation problem with the purpose

* Umut Tosun. Tel.: +0-090-312-2466661/2099 ; fax: +0-090-312-2466660.
E-mail address: utosun@baskent.edu.tr

of minimizing the container re-handling operations at a shipyard[9], travelling salesman, bin-packing, maximum clique, linear ordering, and the graph-partitioning problem are among the interesting applications of the *QAP*.

In its simplest form, the *QAP* is the problem of assigning n facilities to n locations. The cost is calculated as multiplication of flow between the facilities and location distances. Initial cost of placing facilities to the locations can be added to this cost optionally. The objective is to find an allocation such that the total cost of allocating and operating all facilities is minimized. The *QAP* can be formally modeled by using three $n{\times}n$ matrices, A, B, and C.

$$A = (a_{ik}) \tag{1}$$

where a_{ik} is the flow amount from facility i to facility k.

$$B = (b_{jl}) \tag{2}$$

where b_{jl} is the distance from location j to location l.

$$C = (c_{ij}) \tag{3}$$

where c_{ij} is the cost of placing facility i at location j.

The Koopmans-Beckmann form of QAP can be written as:

$$min_{\phi \epsilon S_n}\Big(\sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik} b_{\phi(i)\phi(k)} + \sum_{i=1}^{n} c_{i\phi(i)}\Big) \tag{4}$$

where S_n is the permutation set from 1 through n. The product $\mathrm{a}_{ik}\mathrm{b}_{\phi(i)\phi(k)}$ is the transportation cost caused by settling facility i to location ϕ (i) and facility k to location ϕ (k). In this paper we first give a formal description of the QAP. In Section 2 the related work on QAP is explained. Section 3 introduces the Order-1 crossover and the suggested crossover operators. The environment and the test results obtained with different instances of the QAPLIB are discussed in Section 4 . Finally, Section 5 presents our concluding remarks

2. Related Work

The *QAP* has been studied extensively since it was introduced in 1957. It is proven to be NP-complete[10], so that no polynomial time algorithm is able to exactly solve this problem for larger data sets. Several algorithms have been proposed for both exact and approximate solutions to the problem. Exact algorithms are limited to solving small data sets of the *QAP* with massively parallel computers whereas metaheuristics can provide near-optimal solutions within reasonable optimization times. This property of metaheuristics has made them prominent for solving the *QAP* instances, therefore many researchers have proposed different heuristics or hybrid approaches to solve this problem. In this section, we present a summary of the successful approaches in the literature.

The applicability of the *QAP* to the solution of many different problems has made it the subject of extensive research area for exhaustive and metaheuristic strategies. Small size *QAP* instances are appropriate for exact solutions but the larger instances cannot be solved in reasonable times due to the computational limits. Therefore, metaheuristic approaches have gained a reputation for their ability to produce high-quality solutions within the computational limitations. Simulated Annealing[11],[12], Neural Networks[13], Genetic Algorithms (*GA*s)[14],[15],[16], GRASP[17], Tabu Search (*TS*)[18], and Ant Colony Optimization[19] are some of the well-known metaheuristics that have been successfully applied to the *QAP*.

The solution provided by *TS* procedure is combined with a so-called robust tabu search, *RTS* by Taillard[20], by Misevičius[22] while perturbing the solution via diversification operators. This algorithm effectively explores the symmetric and asymmetric instances given by Taillard from the *QAPLIB*[21]. Hybrid algorithms which exploit *RTS* like sequential metaheuristics, produce high quality solutions in combination with *GA* variants, as shown by Misevičius[22] where two *GA*s are combined with *RTS* based on diversification operators. The first algorithm applies a ruin-and-recreate strategy called *M-GA/TS* and recreates a solution that has been perturbed by the ruin procedure or the crossover. The second algorithm *M-GA/TS-I* perturbs the solutions provided by the *GA* operators by applying a random ruin procedure. Ahuja *et al.*[23] and Drezner[24] have also successfully incorporated *GA* variants into *TS*. The algorithms developed by Drezner[24],[25] perform well especially on instances given by Skorin-Kapov in the *QAPLIB*. The *D-GA/SD* algorithm developed by Drezner[25] implements a crossover operator called a merging process, coupled with a greedy local search that executes swaps until a local optimum is found.

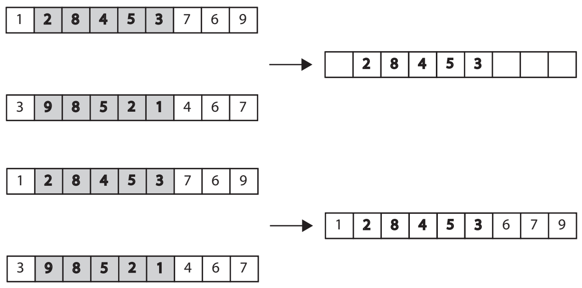

Fig. 1: Steps of Order 1 crossover.

Algorithm 1 Modified Partition Operator for Sorted Crossover

```
partition(array, left, right, pivot) {
    keep the partition with smallest cost;
    pivotValue = array[pivot];
    swap array[pivot] and array[right];
    store = left;
    for i from left to right - 1
        if (array[i] <= pivotValue) {
            swap array[i] and array[store];
            store++;
        }
    swap array[store] and array[right];
    return store;
}
```

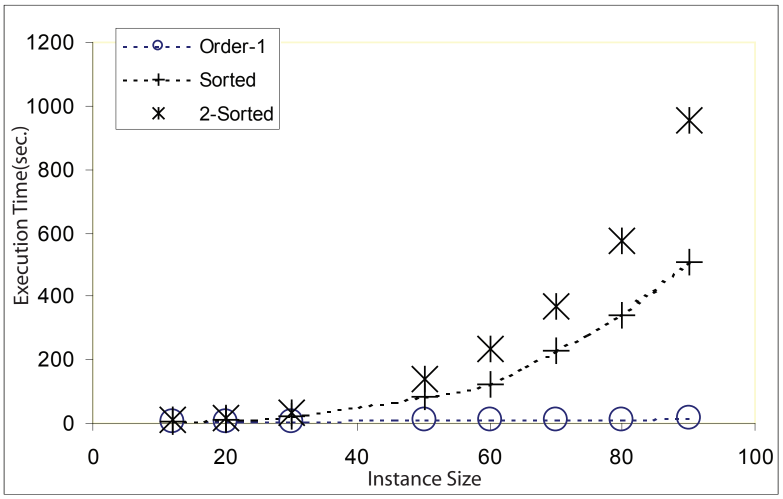

Fig. 2: Instance size vs. time in seconds.

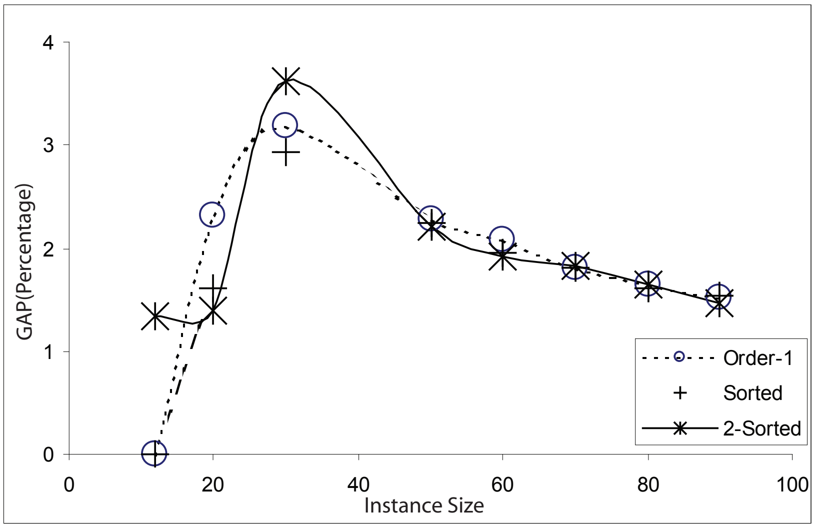

Fig. 3: Instance size vs. Gap btw. optimal solutions.

Table 1: GAP vs. Execution Times for Order-1, Sorted and 2-Sorted crossover.

	FO(GAP)%	Sorted(GAP)%	2Sorted(GAP)%	time1(sec.)	time2(sec.)	time3(sec.)
bur26a	**1,21**	**1,08**	**0,92**	**7,22**	**19,71**	**20,12**
bur26b	**1,30**	**0,93**	**0,65**	**8,60**	**24,99**	**25,37**
bur26c	**1,75**	**1,51**	**1,31**	**8,70**	**18,97**	**25,66**
bur26d	**1,55**	**1,35**	**0,56**	**7,67**	**23,20**	**30,20**
bur26e	**1,15**	**1,08**	**1,08**	**6,99**	**17,46**	**28,86**
bur26f	**1,49**	**0,56**	**0,85**	**7,00**	**17,24**	**28,65**
bur26g	**1,72**	**0,74**	**1,28**	**7,24**	**17,44**	**21,68**
esc16a	**2,94**	**2,94**	**0,00**	**8,42**	**11,38**	**12,41**
esc16b	**0,00**	**0,00**	**0,00**	**9,14**	**10,61**	**13,90**
esc16c	**0,00**	**0,00**	**0,00**	**8,83**	**10,68**	**12,62**
esc16d	**0,00**	**0,00**	**0,00**	**7,30**	**11,57**	**14,86**
esc16e	**0,00**	**0,00**	**0,00**	**7,29**	**11,82**	**11,30**
esc16f	**0,00**	**0,00**	**0,00**	**10,20**	**11,81**	**13,48**
esc16g	**7,69**	**7,69**	**7,69**	**7,43**	**10,48**	**11,97**
esc16h	**0,00**	**0,00**	**0,00**	**6,37**	**9,37**	**11,80**
esc16i	**0,00**	**0,00**	**0,00**	**7,47**	**15,20**	**12,55**
esc16j	**0,00**	**0,00**	**0,00**	**8,79**	**21,70**	**13,49**
esc32c	**0,93**	**0,31**	**0,62**	**7,36**	**45,10**	**41,55**
esc32e	**0,00**	**0,00**	**0,00**	**8,23**	**40,65**	**48,71**
esc32f	**0,00**	**0,00**	**0,00**	**8,03**	**27,97**	**36,26**
esc32g	**0,00**	**0,00**	**0,00**	**8,31**	**24,88**	**36,42**
had12	**0,00**	**0,00**	**1,33**	**6,63**	**8,33**	**10,69**
had14	**1,17**	**0,07**	**0,29**	**6,48**	**10,01**	**9,54**
had16	**1,18**	**0,91**	**0,38**	**6,52**	**10,38**	**10,89**
had18	**1,12**	**0,56**	**0,75**	**7,07**	**11,02**	**13,85**
had20	**2,31**	**1,62**	**1,39**	**6,77**	**11,58**	**15,42**
lipa30a	**3,19**	**2,93**	**3,61**	**7,22**	**21,63**	**32,70**
lipa50a	**2,27**	**2,24**	**2,20**	**8,69**	**85,63**	**141,50**
lipa60a	**2,09**	**1,95**	**1,92**	**10,00**	**121,40**	**233,65**
lipa70a	**1,81**	**1,80**	**1,83**	**10,76**	**229,24**	**365,97**
lipa80a	**1,65**	**1,60**	**1,65**	**12,55**	**340,41**	**574,34**
lipa90a	**1,53**	**1,54**	**1,46**	**16,90**	**508,97**	**954,09**
nug18	**7,56**	**6,32**	**4,97**	**6,15**	**8,90**	**12,06**
Tai15b	**0,95**	**0,77**	**0,80**	**5,98**	**7,04**	**12,02**
Tai25a	**9,84**	**7,61**	**9,82**	**6,40**	**12,54**	**21,19**
Tai64c	**5,31**	**2,42**	**3,56**	**9,57**	**161,69**	**330,23**
Tai80a	**11,44**	**10,94**	**11,42**	**11,49**	**302,33**	**594,03**
Tai100a	**11,02**	**10,62**	**9,55**	**14,22**	**587,06**	**1025,65**
wil50	**7,65**	**7,18**	**7,52**	**7,62**	**75,94**	**127,04**

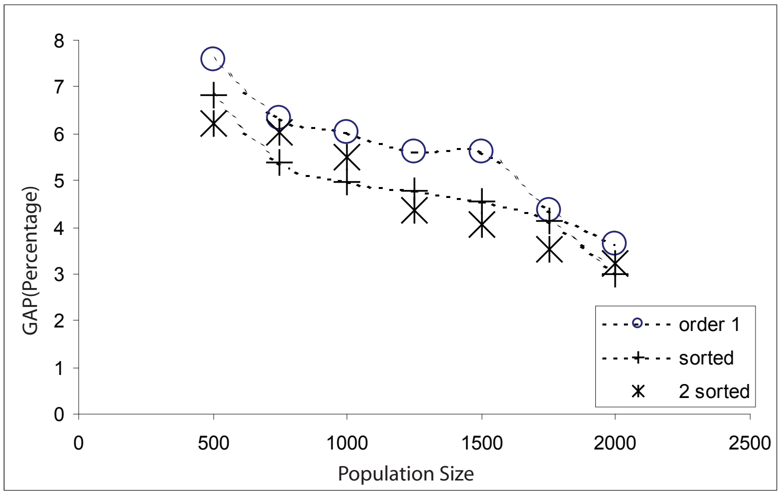

Fig. 4: Population size vs. Gap btw. optimal solution for Nug18.

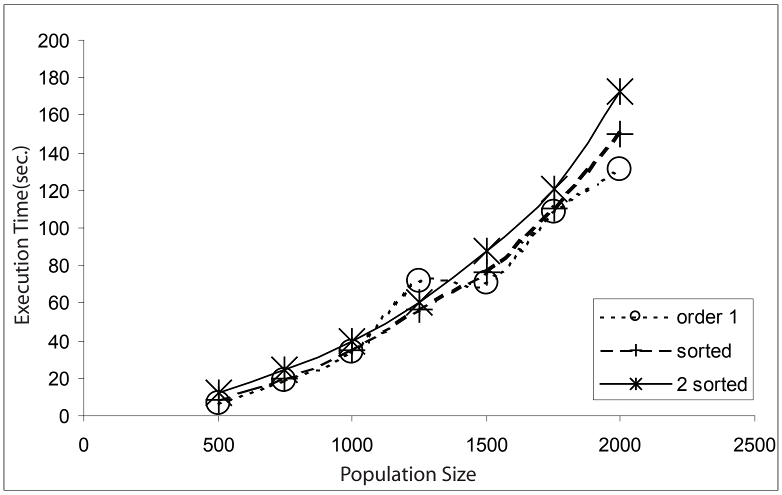

Fig. 5: Population size vs. time in seconds for Nug18

3. Order-1, Sorted and 2-Sorted Crossover Operators

*GA*s generate new individuals by guaranteeing that the best individuals will not be discarded in future generations. In this algorithm, the fittest individuals are copied into the next generations. The crossover operator generates new individuals by recombining the characteristics of parents. For each pair of individuals, the parent chromosomes are split into parts and genes are exchanged to generate new chromosomes. Individuals not subjected to any operation are copied into the next generation.

The idea of Order 1 crossover is to preserve the relative order that the genes occur in the parent chromosomes. An arbitrary part from the first parent is chosen and this part is copied to the first child with the same order as in the first parent. Then, the rest of the genes in the first parent are copied to the first child as in the order of the second parent. The second child is created analogously as shown in Figure 1[26].

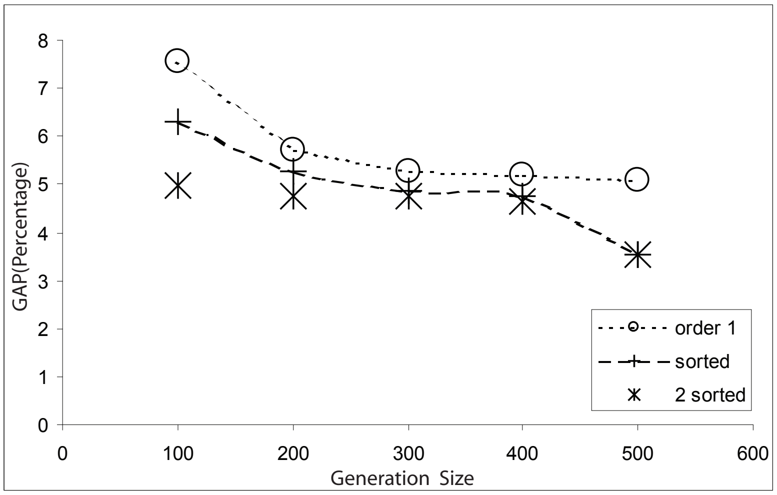

Fig. 6: Generation size vs. Gap btw. optimal solution for Nug18.

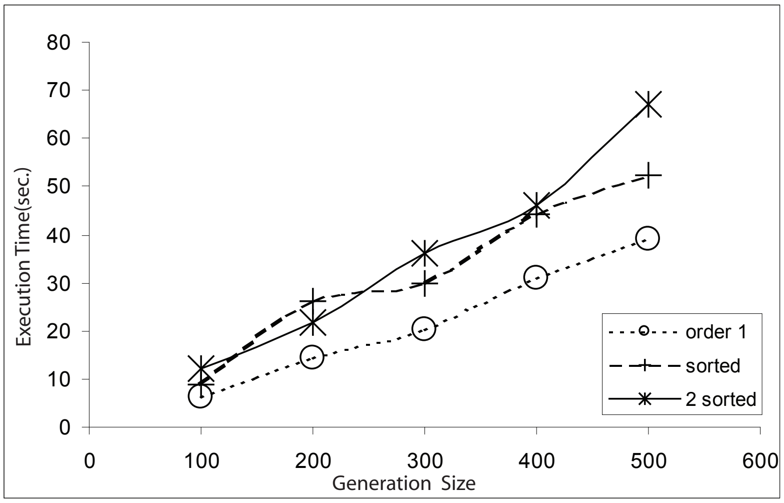

Fig. 7: Generation size vs. time in seconds for Nug18

Sorted Crossover runs the quicksort[27] partitioning algorithm and finds the smallest cost partition generated to crossover with the other chromosome. 2-Sorted crossover is the application of the crossover strategy in two way to both chromosomes. Algorithm 1 shows the modified partitioning algorithm of quicksort[27].

4. Experimental Results

We tested the proposed algorithms through a number of experiments. In each test, one parameter varies while fixing the others. The algorithms are tested by the same test data. Experiments are performed using a 2.21 GHz AMD Athlon (TM) 64x2 dual processor with 2GB RAM and MS Windows 7 (TM) operating system. The implementation language is C++. We reported the best run over 10 consecutive runs. GAP is the percentage gap between the solution found and the optimal solution reported in the QAPLIB. Table 1 shows the results of applying Order-1, Sorted and 2-Sorted crossover operators on some selected instances from the QAPLIB. Figure 2 and Figure 3 show the performance of crossover operators on various instances of size 10 to 100 from the QAPLIB. Sorted Crossover shows better performance than other algorithms for instance sizes smaller than 50. The execution times of the operators are feasible for sizes smaller than 50 either. However, sorted and 2-sorted crossover operators have little or no effect in solution quality for instance sizes larger than 50 and the execution time of the algorithm grows exponentially. This is a fact resulting from the extra partitioning cost of quicksort algorithm on partition generation. 2-Sorted crossover fails to outperform Sorted and Order-1 crossovers because it is elitist and it does not include the diversification of chromosomes in its plan. Figure 4 and Figure 5 show the effect of increasing the population size of genetic algorithm. The algorithm performance decreases steep for population sizes up to 1000. This is meaningful because the genetic algorithm has a breakthrough at some point and it is more difficult to improve the solution from that point. Sorted Crossover performs better than other operators. The execution times are close to linear. Figure 6 and Figure 7 show the performance of the crossover operators when number of generations are increased. Working with larger population sizes have more effect than working with larger generation sizes on the performance of the crossover operators. In our experiments, we used a mutation rate of 0.01 and truncation is used for population selection.

5. Conclusions and Future Work

In this paper, several modifications on Order-1 crossover operator was discussed. Datasets from QAPLIB were used to compare the suggested crossover operators. We used quicksort partitioning to generate a low cost chromosome and produce the offsprings. The partitioning algorithm may be applied to both of the chromosomes or to only one chromosome. Sorted crossover shows better results than 2-sorted crossover because most of the time 2-sorted chromosome has an elitist behavior. Even though, sorted and 2-sorted crossover show good performance results for instance sizes smaller than 50, they run in higher execution times for larger instances. Sorted crossover is very promising and observed to be the optimal algorithm running on meaningful times. However, partitioning adds a considerable cost to order-1 crossover. Hybrid algorithm performance of the suggested crossover methods and different modifications that run in low execution times may be of interest in the future.

References

1. D. W. Pentico, "Assignment problems: a golden anniversary survey," *European Journal of Operational Research*, vol. 176, no. 4, pp. 774-793, 2007.
2. R. E. Burkard and E. Cela, "Linear assignment problems and extensions," In Pardalos, P. and Du, D.-Z. (eds), Handbook of Combinatorial Optimization, *Kluwer Academic Publishers*, Supplement vol. A, pp. 75-149, 1999.
3. T. C. Koopmans and M. J Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, no. 1, pp. 53-76, 1955.
4. L. Steinberg, "The backboard wiring problem: A placement algorithm," *SIAM Review*, vol.3, no. 1, pp. 37-50, 1961.
5. D. F. Rossin, M. C. Springer, and B. D. Klein, "New complexity measures for the facility layout problem: An empirical study using traditional and neural network analysis," *Computers and Industrial Engineering*, vol. 36, no. 3, pp. 585-602, 1999.
6. G. F. Pfister, *In Search of Clusters (2nd Edition)*. Prentice Hall, 1998.
7. M. H. Lim, Y. Yuan, and S. Omatu, "Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem," *Computational Optimization and Applications*, vol. 15, no. 3, pp. 249-268, 2000.
8. R. K. Adl and S. M. T. R Rankoohi, "A new ant colony optimization based algorithm for data allocation problem in distributed databases," *Knowledge Information Systems*, vol. 20, no. 3, pp. 349-373, 2009.
9. J-F. Cordeau, M. Gaudioso, G. Laporte, L. Moccia, "The service allocation problem at the Gioia Tauro Maritime Terminal," *European Journal of Operational Research*, vol. 176, no. 2, pp. 1167-1184, 2007.
10. S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the Association for the Computing Machinery*, vol. 23, no. 3, pp. 555-565, 1976.
11. M. R. Wilhelm, "Solving quadratic assignment problems by simulated annealing," *IIE Transactions*, vol. 19, no. 1, pp. 107-119, 1987.
12. D. T. Connolly, "An improved annealing scheme for the QAP," *European Journal of Operational Research*, vol. 46, no. 1, pp. 93-100, 1990.
13. C. Calzon-Bousono, "The hopfield neural network applied to the quadratic assignment problem," *Neural Computing and Applications*, vol. 3, no. 2, pp. 64-72, 1995.
14. J. P. Cohoon and W. D. Paris, "Genetic placement," *IEEE Transactions on Computer-Aided Design*, vol. 6, no. 6, pp. 956-964, 1987.
15. D. M. Tate and A. E. Smith, "A genetic approach to the quadratic assignment problem," *Computers and Operations Research*, vol. 22, no. 1, pp. 73-83, 1995.
16. U. Tosun, A. Cosar, T. Dokeroglu, "A Robust Island Parallel Genetic Algorithm for the Quadratic Assignment Problem", International Journal of Production Research, vol 51, pg 4117, 2013.

17. Y. Li, P. M. Pardalos, and M. G. C. Resende, ”A greedy randomized adaptive search procedure for the quadratic assignment problem,” Quadratic assignment and related problems, P. M. Pardalos and H. Wolkowicz, eds., *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, vol. 16, pp. 237-261, 1994.
18. R. Battiti and G. Tecchiolli, ”The reactive tabu search,” *ORSA Journal on Computing* , vol. 6, no. 2, pp. 126-140, 1994.
19. L. M. Gambardella, E. D. Taillard, and M. Dorigo, ”Ant colonies for the quadratic assignment problem,” *Journal of the Operational Research Society*, vol. 50, no. 2, pp. 167-176, 1999.
20. E. Taillard, ”Robust taboo search for the quadratic assignment problem,” *Parallel Computing*, vol. 17, no. 4-5, pp. 443-455, 1991.
21. R. E. Burkard, S. E. Karisch, and F. Rendl, ”QAPLIB a quadratic assignment problem library,” *European Journal of Operational Research*, vol. 55, no. 1, pp. 115-119, 1991.
22. A. Misevičius, ”A tabu search algorithm for the quadratic assignment problem,” *Computational Optimization and Applications*, vol. 30, no. 1, pp. 95-111, 2005.
23. R. K. Ahuja, and J. B. Orlin, A. Tiwari, ”A greedy genetic algorithm for the Quadratic Assignment Problem,” *Computers and Operations Research*, vol. 27, no. 10, 917-934, 2000.
24. Z. Drezner, ”The extended concentric tabu for the quadratic assignment problem,” *European Journal of Operational Research*, vol. 160, no. 2, pp. 416-422, 2005
25. Z. Drezner, ”A new genetic algorithm for the quadratic assignment problem,” *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 320-330, 2003.
26. Eiben, A.E. and Smith, J.E., 2003. *Introduction to evolutionary computing*, Springer.
27. C.A.R. Hoare, Quicksort, Computer Journal, 5(1):10-15, 1962.