INTERNATIONAL JOURNAL OF ELECTROINCS; MECHANICAL and MECHATRONICS ENGINEERING Vol.3 Num 2 pp.(557-570)

ANALYSIS AND PERFORMANCE MEASUREMENT OF EXISTING SOLUTION METHODS OF QUADRATIC ASSIGNMENT PROBLEM

Morteza KARAMI[1], Sadegh NIROOMAND[2], Nima MIRZAEI[2], Bela VIZVARI[1]

[1] *Department of Industrial Engineering, Eastern Mediterranean University, North Cyprus, Turkey*
Email: m.morteza01@yahoo.com
Email: bela.vizvari@emu.edu.tr

[2] *Department of Industrial Engineering, Istanbul Aydin University, Istanbul, Turkey*
Email: sadeghniroomand@aydin.edu.tr
Email: nimamirzaei@aydin.edu.tr

Abstract- *Quadratic Assignment Problem (QAP) is known as one of the most difficult combinatorial optimization problems that is classified in the category of NP-hard problems. Quadratic Assignment Problem Library (QAPLIB) is a full database of QAPs which contains several problems from different authors and different sizes. Many exact and meta-heuristic solution methods have been introduced to solve QAP. In this study we focus on previously introduced solution methods of QAP e.g. Branch and Bound (B&B), Simulated Annealing (SA) Algorithm, Greedy Randomized Adaptive Search Procedure (GRASP) for dense and sparse QAPs. The codes of FORTRAN for these methods were downloaded from QAPLIB. All problems of QAPLIB were solved by the above-mentioned methods. Several results were obtained from the computational experiments part. The Results show that the Branch and Bound method is able to introduce a feasible solution for all problems while Simulated Annealing Algorithm and GRASP methods are not able to find any solution for some problems. On the other hand, Simulated Annealing and GRASP methods have shorter run time comparing to the Branch and Bound method. In addition, the performance of the methods on the objective function value is discussed.*

Keywords: *Quadratic Assignment Problem, Quadratic Assignment Problem Library, Branch and Bound, Simulated Annealing, Greedy Randomized Adaptive Search Procedure*

1. INTRODUCTION

The aim of Quadratic Assignment Problem (QAP) is to assign a set of given tasks (say set 1) to another set of given tasks (say set 2) with a given assignment cost/benefit matrix in order to minimize/maximize total cost/benefit of the assignments. This problem is restricted to assign each task of set 1 to only one task of set 2 and also only one task of set 1 can be assigned to each task of set 2 by the use of binary variables. QAP has many applications in the real world where, the aim is to assign a set of jobs (set 1) to a set of machines/workers (set 2) with a given matrix including cost of assignment of each job to each machine/worker in order to decrease the total assignment cost. The aim is to assign a set of facilities (set 1) to a set of given and fixed locations (set 2) with given flow matrix of facilities and distance matrix of locations in order to decrease the cost of assignment that is calculated by multiplying the distance of a pair of location and the flow between their related facilities for all possible pairs of locations. In addition to the above-mentioned assignment problems, QAP may be used as a mathematical formulation for the placement problem of interconnected electronic components onto an integrated circuit board or on an electronic microchip, which is a part of computer aided design in the electronics industry. QAP is an NP-hard problem. The Branch and Bound (B&B) method is a well-known exact method for solving QAPs. The method is used in most of optimization software e.g. Lingo, Xpress, Cplex, etc. This method can be more effective for solving QAPs where some linearization techniques are used to linearize the quadratic terms of objective function. The methods were effective to reduce the running time of the problem (see He *et al*. (2012))

1.1 QAP applications

Koopmans and Beckmann in 1975 were the first proposers of quadratic assignment problems as a mathematical model connected to the location of economic activities. After that it was shown in difference practical application: (Steinberg, 1961) used the QAP to show the optimal placement of computer element on the backboard wiring; (Dickey & Hopkins, 1972) done the Campus building arrangement by using QAP. (Francis & White, 1974) used QAP for assign some facilities (police posts, supermarkets, schools and so on) to best location in order to have the best service. The QAP is used to find the best place for typewriter keyboard and control panel by (Pollatschek, Gershoni, & Radday, 1976). Quadratic assignment, as a general data analysis strategy, is defined by (Hubert & Schulz, 1976). (Elshafei, 1977) utilized the quadratic assignment problem for the Hospital planning. (Krarup & Pruzan, 1978) applied computer aided layout design to archeology. also (Rabak & Sichman, 2003) and (Miranda *et al*. 2005) studied the best place of electronic elements. (Wess & Zeithofer, 2004) studied On the phase coupling problem between data memory layout generation and address pointer assignment. Generally, because of more benefit and importance of QAP in different industry and place, a lot of papers have pressed and new techniques for these problems created until now and will be continue in future.

1.2 Resolution algorithms for solving QAP

Two general algorithms usually are used for solving optimization problems and specially QAP. The first one is exact algorithm and the second algorithm is heuristic. The most important strategies of these two methods can be explained in a short way as follows. The rest of the algorithms were not used in this project and mentioned only in order to show the vast application of QAP. The following are some exact algorithm that is used for solving QAP in optimization problem.

- **Branch-and-Bound algorithm (B&B*)***: The B&B is one of the well-known and most frequently used methods for solving this kind of optimization problems.
- **Dynamic programming algorithm**: This algorithm is used for some special instances that the flow matrix (matrix B) is the adjacency matrix of a tree. (Christofides & Benavent, 1989) were the first introducers that studied and used this algorithm to the relaxed instances. After that this algorithm was improved and (Urban, 1998) used the framework of dynamic programming to obtain an optimal solution procedure.
- ***Cutting plane algorithm*:** Bazaraa and Sherali in 1979 were the first introducer of this algorithm that at the beginning did not give a good result. This algorithm only used to small size of problems by (Kaufman & Broeckx, 1978) and (Burkard & Bonniger, 1983). In two decades later for solving on computer motherboard design problem used Bender decomposition method by (Miranda, Luna, Mateus, & Ferreira, 2005).
- **Branch-and-cut algorithm:** It is a modified version of the B&B idea. First it was applied by (Padberg & Rinaldi, 1991) for solving symmetric matrices. (Junger & Kaibel, 2000, 2001a,b) and (Blanchard, Elloumi, Faye, & Wicker, 2003) were some of the researchers that improved and applied this algorithm.

1.3 Heuristic and metaheuristics algorithms

The heuristic method does not assure that the best solution achieved is optimal or not. This algorithm is classified in three parts: The first one is a constructive method that first time proposed by (Gilmore, 1962) and in the future developed and used by some other researchers as (Arkin, Hassin, & Sviridenko, 2001) (Gutin & Yeo, 2002). The second class is a limited enumeration that assures the obtain solution value is optimum if that obtain value go to the end of the enumerative procedure. The third and last class of heuristic algorithms that includes most of the Quadratic Assignment Problems is improvement methods. The procedure of this method is parallel with the local search method, and it starts with a feasible solution and attempt to make it better (Mills, Tsang, & Ford, 2003). The heuristic algorithms were just used for particular numerical problem before 1990s. However, at the end of 1990s some general algorithms were introduced and it is named metaheuristics. Several of most important techniques that were used in assignment problem field are as follow:

- **Simulated Annealing algorithms(SA)**: This is one of the good and useful methods for solving NP hard problems, like as QAP. The SA method is a metaheuristic optimization method to solve combinatorial optimization problems. This method was introduced by (Kirkpatrick et al. 1983).
- ***Genetic Algorithms***: for QAP classified in two phases, the first phase is finding the starting population by saving the best solution of each iteration. And the second phase is using Genetic Algorithms to achieve the optimal solution. For more information about

the genetic procedure can see the published paper about this algorithm.

- **Scatter search**: Glover in 1977 was the first introducer of this method for integer programming instances. This method also contains two phases, the first phase is investigate a pleasant solution and called initial phase and the evolutionary phase (the second phase) is continue and repeat this process until achieve to a stop criterion. (Cung, Mautor, Michelon, & Tavares, 1997) studied and applied this method for the QAP instances.
- **Tabu search algorithm**: This algorithm is used to integer programming problems. Glover in (1989 a,b) was the researcher who introduced this algorithm.
- **Greedy Randomized Adaptive Search Procedure (GRASP)**: It is a good method that mixes lot of favorable properties of other heuristics. In GRASP can select number of iterations as each iteration includes two steps, a first one is construction and the second is local search. Each iteration gives a special solution and the best solution from all iterations is selected for the final solution of the main problem (Pardalos & Crouse, 1989).

2. SELECTED METHOD FOR SOLVING QAP

2.1 Branch and bound

In this method the solution status at any point is described by an unexplored subset and the best solution which founded so far. Only one subset exist initially (say complete solution space) and the best solution value obtained so far is infinity. The nodes in search tree represent unexplored subspaces and they contain the root initially. Three main step of iteration can be classified as: Choosing the node, computing the bound and final branching. Figure 1; represent the first step and initial condition (Clausen, 1999). The procedure continues with choosing the next node. The first action will be branching if sub-problem selected according to the bound value. For each subspace which had single solution we keep this solution and then make comparison with the current solution and save the best one. If it does not contain any single solution, bounding function calculation and comparison with current best solution should be made. The subspace can be discarded completely if it is founded that the optimal solution cannot be reached through the subspace, otherwise it must be saved (Clausen, 1999). When the solution space explored completely the procedure ends and what saved in "current best" will be the optimal solution.

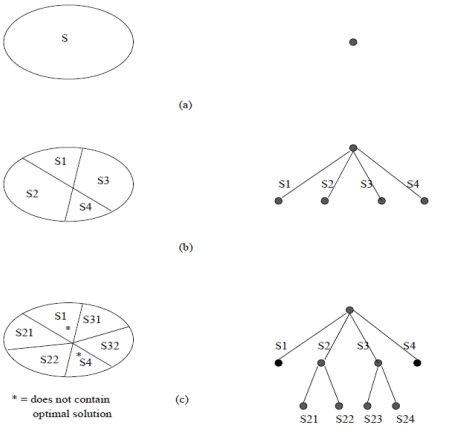

Figure 1: Illustration of the search space of B&B

Branch and Bound method for QAP classified in three steps:

- Single assignment problem ((Gilmore, 1962) , (Lawler, 1963))
- Pair assignment algorithms ((Gavett & Plyter, 1994), (Land, 1963), (Nugent, Vollmann, & Ruml, 1969))
- Relative positioning algorithm ((Mirchandani & Obata, 1979))

Roucairol in 1987 and Pardalos were the first researchers who introduced B&B algorithms for quadratic assignment problem, but they just worked on instances of size $n \leq 15$. This algorithm was developed further on in one decade later by many researchers, such as (Gendronu & Grainic, 1994), (Feo & Resende, 1995) and (Pardalos, Resende and Ramakrishnan, 1995).

2.2 Simulated annealing (SA)

This algorithm has been applied to many layout problems e.g. (Peng, Huanchen, & Dongme, 1996) and (Jingwei, Ting, Husheng, Jinlin, & Ming, 2012), *etc*. The SA algorithm is based on randomization techniques and works as a probabilistic local search method. Generally the SA applies an iterative improvement algorithm to a predetermined initial solution and its objective function value. The SA with continues stage can be used to find and improve the local optimization algorithm. Also we can achieve to the local optimization algorithm by generating of all solutions in the neighborhood of the current solution. Local Search evaluates the neighbor according to the optimization criterion and selects it if it is better than the current solution, otherwise rejects the neighbor. This procedure continues until that the all or maximum number of the trials was checked and is not able to improve the solution. The SA is differ from local

optimization, in SA we can accept solution **Á** if it is better than solution **A** or worse than **A**. It depends on Boltzmann's law that uses to determine this acceptance probability that is shown as follow (Singh & Sharma, 2006):

$$p(accept) = e^{-\Delta z/bt}$$

The objective function value of the solutions obtained by SA algorithm usually is close to optimal value. The Simulated Annealing algorithm consists of the following steps:

Step 1: For first solution of SA we choose initial solution 'i' randomly.
Step 2: An initial temperature should be chosen (T_i>0).
Step 3: Then we should select the temperature updating function i.e. cooling (or annealing) schedule.
Step 4: The epoch length function should be selected.
Step 5: Put temperature change counter (t) and epoch length counter (l) equal zero.
Step 6: By replacing two facilities compute solution Á in the neighborhood of **A**.
Step 7: Compute $\Delta z = z(Á) - z(\boldsymbol{A})$
Step 8: If $\Delta z \geq 0$ go to step 9, otherwise replace a with **Á** and go to step 10.
Step 9: If random (0 ,1) < $\exp(-\Delta z/bt)$ then **Á**=**A**
Step 10: 3 last step (7 to 9) repeat until l=Q (the maximum Q (the number of trial) for which the temperature is 't')
Step 11: Generate and find the next temperature according to step 3 so that the function will be change and again repeat step 6 till 9 for the new temperature.
Step 12: We should do all of steps again until the stopping criteria becomes true.

In SA procedure b is a Botzmann's constant and parameter, t is called temperature that can change according to some annealing schedule and also $T_I < t < T_F$ (T_I is the initial and T_F is final temperatures respectively). For using SA method four factors must be selected: Initial temperature, Epoch length, Annealing (Cooling) schedule and finally termination criterion. (Singh & Sharma, 2006)

2.2.1 Initial temperature

A great initial temperature introduced by (Kirkpatrick, Gelatt Jr, & Vecchi, 1983) in optimizing by simulated annealing article that with probability 8% most of the solutions are accepted at the first stage of the SA procedure.

2.2.2 Epoch length

If we assume N_k be the epoch length parameter (i.e. the amount of trials to be done with the same temperature value). Some functions that refer to them are as follows:

Constant function: N_k = constant (k = 0,1,2,…,Q)
Arithmetic function: : $N_k = N_{k-1}$ + constant (k = 0,1,2,…,Q)
Geometric function: $N_k = \frac{N_{k-1}}{a}$ (a<1 and constant, k = 0,1,2,…,Q)
Exponential function: $N_k = \frac{constant}{\log(T_k)}$ (k = 0,1,2,…,Q)
Logarithmic function: $N_k = (N_{k-1})^{\frac{1}{a}}$ (a<1 and constant, k = 0,1,2,…,Q)

2.2.3 Cooling schedule

A lot of functions for updating temperature are available in literature, but sum of the most important ones are as follows:

Arithmetic function: : $t_{k+1} = t_k -$ constant (k = 0,1,2,…,Q)
Geometric function: $t_{k+1} = \alpha . t_k$ (α<1, $t_0 = T_I(constant)$, k = 0,1,2,…,Q)
Inverse function: $t_{k+1} = \frac{t_k}{1+\alpha . t_k}$ (α < t_0 , $t_0 = T_I(constant)$, $t_k = \frac{constant}{t_{k+1}}$, k=0,1,2,…,Q)
Logarithmic function: $t_k = \frac{constant}{\log(t_{k+2})}$ (k = 0,1,2,…,Q)

2.2.4 Termination criterion

In the literature lot of tests available and explained that some of them are as follows:
If we can't find any improvement in a certain amount of iterations;
If all of the iterations have been done;
If the previously defined number of acceptance for a given number of trials has not been obtained;
If we reached the target temperature.

2.3 GRASP method

The GRASP procedure is shown on Figure 2: (Pardalos & Crouse, 1989)

```
procedure grasp()
1       InputInstance();
2       do stopping criterion not satisfied →
3           ConstructGreedyRandomizedSolution(solution);
4           LocalSearch(solution);
5           SaveSolution(solution,bestsolution);
6       od;
7       return(bestsolution)
end grasp;
```

Figure 2: General case for GRASP pseudo-code

In row number one we have inputs that consists two matrices, the flow matrix (F) and distance matrix (D). The stopping criterion in line 2 is a maximum number of iteration that we select or a solution value that we are satisfied or anything else. Line 3 is the construction step (shown in Figure 3) and line 4 is the local search step of GRASP procedure. In line 5 records the best solution until now and continue (Pardalos & Crouse, 1989). Figure 3 illustrate the summary of construction step. In this figure line 1 means that the construction step start with empty solution. In line 3 create a Restricted Candidate list (RCL). RCL means a list of best components, from that a selection will be created. The selecting random elements from RCL list is done in line 4. In line 5 the new solution is considered as the last solutions set with added s. The last elements (without added s) are computed in the greedy function, where the loop starting with a new RCL will be constructed.

```
procedure ConstructSolution(solution)
1       solution={};
2       do Solution is not complete →
3           MakeRCL(RCL);
4           s=SelectRandomElement(RCL);
5           solution=solution ∪{s};
6           AdaptGreedy(s,RCL);
7       od;
8       return(solution)
end ConstructSolution;
```

Figure 3: GRASP construction step pseudo-code

The construction step can be implemented in two ways, one of them is sparse procedure and another one is dense procedure. Both of these procedures have similar function, but the most important thing is that the sparse procedure is considerably faster rather than dense procedure. This fasting is perceptible in our computations.

2.3.1 Initial construction phase for QAP

Each QAP instance consists of two $n \times n$ matrices which are the flow matrix F= (f_{ij}) and the distance matrix D= (d_{ij}). $f_{ij}d_{kl}$ is the transportation cost between facilities i and j if they are assigned to locations k and l. α and β are assumed the candidate list restriction parameters. Then the distance matrix elements classified in increasing order and flow matrix element classified in decreasing order that shown as follow:

$$d_{k_1 l_1} \leq d_{k_2 l_2} \leq \ldots \leq d_{k_m l_m}$$
$$f_{i_1 j_1} \geq f_{i_2 j_2} \geq \ldots \geq f_{i_m j_m}$$

If $\lfloor \beta_m \rfloor$ is the minimum of d_{kl} elements and also $\lfloor \beta_m \rfloor$ is the maximum of f_{ij} elements, the corresponding cost is as follows:

$f_{i_1 j_1} d_{k_1 l_1}, f_{i_2 j_2} d_{k_2 l_2}, \ldots, f_{i_{\lfloor \beta_m \rfloor} j_{\lfloor \beta_m \rfloor}} d_{k_{\lfloor \beta_m \rfloor} l_{\lfloor \beta_m \rfloor}}$

These costs are classified in increasing order and the cost corresponding to some couple assignment shown with each element. In in this phase according to two couple assignment, two selection elements take from the previous cost set. For complete implementation details about this phase can refer to (Li, Pardalos, & Resende, 1994).

2.3.2 Local search for QAP

In this step the current solution (s) is improved by investigating the neighborhood N(s) of that solution. QAP include n! permutation. First of all assume $\delta(p,q) = \{i | p(i) \neq q(i)\}$ is the difference between permutation p and permutation q and $d(p,q) = |\delta(p,q)|$ is the distance of this this two permutations. For different problems, many different ways are available to construct N(s). Some of the general ones are as follows (Li (1992)):

The reasonable neighborhood size: could be possible to investigate in the rational number of calculation.
Large variance in the neighborhood: show that the maximum distance of all of the permutations is large if a special neighborhood conclude all of the permutations $p_1, p_2, \ldots, p_n$. Connectivity in the neighborhood: it means that the sequential of permutation $\{p_k\}$ with small $\delta(p_k, p_{k+1})$ are available while we moving from permutation $p_i \; to \; p_j \; (k = i, i+1, \ldots, j-1)$.
The k-exchange neighborhood is used for GRASP local search, which is as follows:
$N_k(p) = \{q | d(p,q) \leq k, 2 \leq k \leq n\}$.
The initial permutation p is used to start the k-exchange local search, and then all of the permutations generated by exchanging of k between each other. The local optimum recorded according to the best permutation. The size of this neighborhood is p_k^n. If $k = 2$ then the neighbors are as follows:

$$\begin{array}{ll} m = 1 & q = (2,1,3,\ldots,n-2,n-1,n) \\ m = 2 & q = (3,2,1,\ldots,n-2,n-1,n) \\ m = p_k^n - 1 & q = (1,2,3,\ldots,n-1,n-2,n) \\ m = p_k^n & q = (1,2,3,\ldots,n-2,n,n-1) \end{array}$$

In looking for the neighborhood, the k-exchange local search processing mentioned above ceases when a better solution is achieved, and starts penetrating in the neighborhood until the most valued solution is found at the same manner. The First Decrement search is comparing to Complete Enumeration search, but

Complete Enumeration checks all of the solution in neighborhood, and this procedure continues for the best outcome is suggested as before. However in both case a neighborhood completely search (according to p_k^n solution). But in GRASP procedure generally the initial permutation is in the good condition and the local search just search in few percent of the neighborhood (Pardalos & Crouse, 1989).

3. EXPERIMENTATION

3.1 Quadratic assignment problem library

In order to create a standard test set for QAP, in 1991 QAPLIB was established which is accessible for all researchers and all QAP instances that were available to the authors at that time, are included in it. Following the huge positive feedback and continuing demands from scientific community in 1994 Bukard, Karisch and Rendl provided a major update which was even accessible through anonymous ftp as well. Many new problem instances which were generated by researchers for their own testing purposed, were included in that update. In April 1996 QAPLIB was updated again. On one hand this update reflected on the big changes in electronic communication, which means, QAPLIB became a World Wide Web site, the QAPLIB Home Page. On the other hand, research activities around QAP were increased so much that another update was necessary and some recent (at that time) dissertations were added to the library. Some of the test instances were not solved optimally before, so another update released in June 2000 which contained new test instances and the optimal solutions for non-optimally solved old problems. The next update was in January 2002 that again consists of new test instances and improvements on the best known feasible solutions, especially test the instances did not achieve to the optimal solution yet.

3.2 Problem instances

In this part we discuss the problem instances that are available in the QAPLIB. The size of instances are $12 \leq n \leq 256$, and four largest ones of size 128, 150(two instances) and 256. Instances of size $n < 12$ are not considered in the QAPLIB, because this size of problems can be solved so easily even without using any software. The problems that solved in the QAPLIB consists different parts:
n: is the size of the instance
A and B: are distance and flow matrices
P: is a corresponding permutation
Sol: is an objective function value

Also the solution of the problems classified to two groups:

1) The problem instances where optimality is. Their optimal permutation is also provided.
2) The problem instances where optimality is not achieved. Their best known feasible solution and the gap relative to the best known lower bound are also provided.

This section presents certain numerical results for all of the instances in the QAPLIB (134 instances) were calculated with 4 different algorithms which have been explained in the previous section extensively. The calculations were made by FORTRAN software which is available in QAPLIB home page and FORTRAN codes were modified because the original codes was not appropriate for solving general sizes in the instances so the new codes supported different input size of the matrices. Maximum number of iterations for GRASP method was set to 100 times and the parameters $\alpha = 0.25$ and $\beta = 0.5$ were initialized. 12 desktop computers with processors with Pentium® D 3.00 GHz and 960 Mb RAM with windows XP 32 bit as the operating system, were connected to a network and all runtimes correspond to the parallelized processors. Summary of all the experiments with 4 mentioned methods and the existing results in the literature are shown and compared in Table 1 in Appendix.

This table's columns, contains five parts. The data from first part was taken form QAPLIB home page and the other parts are the results obtained from the algorithms. The first part contains 3 columns. First column is the name of instances, which is abbreviation of author and size of the instance. The second column shows the feasible solution for all the instances and the ones that reached to the optimal solution were marked with (Opt) in gap column and the rest are the best known solution. The gaps in between best known solution and the currently known best lower bound were shown in column number 3. The other 4 parts of this table contain 3 columns each. The first one is the obtained feasible solution and the second column shows the runtime of the solution and the third column shows the Performance ratio. The best value among these four algorithms is bolded. The maximum runtime was set to 1 week and if during this 1 week the solution algorithm did not stop, it will drop the process and record the best know solution obtained by the algorithm(if there was not any solution, the square of the feasible solution marked by (—)) and the runtime of these processes are shown by (1w). The value of performance ratio is obtained by dividing the obtained value with mentioned method by the best known value in the QAPLIB web page. If this value is less than one, it means that the obtained value found is better than the value in the QAPLIB web page. Otherwise, the values

which are closer to one are more reliable. In some of the instances (for example all part b of the Taillard instances), the volume and amount of data matrices and the solution were overclocked and in these cases, the compiler was not able to solve the instance. This problem mostly happens in a few of instances that were solved by GRASP algorithm. For example all of the Burkard instances, the Simulated Annealing method and both of GRASP algorithms can't find any solutions. The reason of this case is that the flow and distance matrices of Burkard instances are not symmetric and solving this kind of problem is difficult. The instances that faced this situation are marked with (N) and also for this kind of problem there is not any runtime and performance ratio. According to the results obtained in this research, about 63% of the solutions in Branch and Bound Algorithm (B&B) were equal or closest to QAPLIB solutions which compared to the other algorithms is the highest percentage. It shows that the accuracy and efficiency of Branch and Bound algorithm is at least twice compared to the other algorithms. The instances which were solved by B&B method reached the objective function value.

The reason for this is that unlike the other methods, B&B method is an exact algorithm and although it takes more time to solve the instances, it will give the objective function value. Based on the average runtimes of each algorithm that led to a feasible solution without taking the instances that did not attain a feasible solution in one week and stopped into the consideration, although the best solutions were obtained by Branch and Bound algorithm, but this algorithm is dependent to a long runtime and has the longest runtime for solving the instances among the other algorithms. The average runtime for this algorithm is more than 265 minutes per instance. This amount of runtime can effect to decision for select the method for QAP. Although the fastest solutions for instances are obtained by Simulated Annealing (with the average run time 0.3 second, but only 20% of the instances reach to best feasible solution by this algorithm. Both types of Grasp algorithms (Dense and Sparse) with the average runtime of 3.60 and 3.53 seconds and 35% and 33% of best feasible solutions, are of the most efficient methods for solving these problems. However, for all Bur and Lipa instances, because of the high volume of data, they do not have a good performance.

4. DISCUSSION OF THE RESULT AND CONCLUSION

The purpose of this research is to illustrate which of these four algorithms mentioned in the previous parts are more suitable for analyzing QABLIB's instances with respect to runtime. According to the feasible solution that recorded in table 1, it can be concluded that, if the runtime is not a priority, the Branch and Bound algorithm is still one of the best algorithms for solving these kinds of optimization problems, but there is no guarantee when is the optimal solution achieve. And if the classification of these algorithms is made with respect to runtimes, the best method is Simulated Annealing with the average runtime of 0.3 seconds, but because this method is a heuristic method, there is no guarantee to reach the objective value. The solution time can be extremely long for large problem instances. Therefore it is possible that within a week the optimal solution is not achieved. Thus the B&B algorithm can be understood not only exact but heuristic method as well. Obviously if a more powerful processor and greater volume of Ram is used for this research, it will lead to a better runtimes.

5. REFERENCES

[1] Arkin, E., Hassin, R., & Sviridenko, M. (2001). Approximating the maximum quadratic assignment problem. *Information Processing, 77*, 13-16.

[2] Bazaraa, M., & Sherali, H. (1979). New approaches for solving the quadratic assignment problem. *Operation Research Verfahren, 32*, 29-46.

[3] Blanchard, A., Elloumi, S., Faye, A., & Wicker, N. (2003). A cutting algorithmfor the quadratic assignment problem. *INFOR, 41*, 35-49.

[4] Burkard, R., & Bonniger, T. (1983). A heuristic for quadratic Boolean programs with applications to quadratic assignment problems. *European Journal of Operation Research, 13*, 374-386.

[5] Christofides, N., & Benavent, E. (1989). An exact algorithm for the quadratic assignment problem. *37*, 760-768.

[6] Clausen, J. (1999, March 12). Branch and Bound Algorithms Principles and Examples. *Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100*.

[7] Cung, V.-D., Mautor, T., Michelon, P., & Tavares, A. (1997). A scatter search based approach for the quadratic assignment problem. *Proceedings of IEEE International Conference on Evolutionary Computation*, 165-169.

[8] Dickey, J., & Hopkins, J. (1972). Campus building arrangement using Topaz. *Transportation Research, 6*, 59-68.

[9] Drezner, Z., & Marcoulides, G. (2003). A distance-based selection of parents in genetic algorithms.Kluwer Academic Publishers. 257-258.

[10] Elshafei, A. (1977). Hospital layout as a quadratic assignment problem. *Operations Research Quarterly, 28(1)*, 167-179.
[11] Feo, T., & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*, 109-133.
[12] Francis, R., & White, J. (1974). Facility Layout and Location: An Analytical Approach. *Prentice-Hall, Englewood Cliffs, NJ*.
[13] Gavett, J., & Plyter, M. (1994). The optimal assignment of facilities to locations by Branch and Bound. *Operation Research, 42*, 860-878.
[14] Gendronu, B., & Grainic, T. (1994). Parallel branch and bound algorithms: Survey and synthesis. *Technical report, Center for Research on Transportation*.
[15] Gilmore, P. (1962). Optimal and suboptimal algorithms for the quadratic assignment problem. *SIAM Journal on Applied, 10*, 305-313.
[16] Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sience, 8*, 156-166.
[17] Gutin, G., & Yeo, A. (2002). Polynomial approximation algorithms for TSP and QAP with a factorial domination number. *Discrete Applied Mathematics, 119*, 107–116.
[18] Hubert, L., & Schulz, J. (1976). Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical Psychology, 29*, 190-241.
[19] Jingwei, Z., Ting, R., Husheng, F., Jinlin, Z., & Ming, L. (2012). Simulated annealing ant colony algorithm for Quadratic Assignment Problem. *International Conference on Natural Computation*, 789-793.
[20] Junger, M., & Kaibel, V. (2000). On the SQAP-polytope. *SIAM journal on Optimization, 11*(2), 444-463.
[21] Junger, M., & Kaibel, V. (2001). The QAP-polytope and the star transformation. *Discrete Applied Mathematics, 111*(3), 283-306.
[22] Kaufman, L., & Broeckx, F. (1978). An algorithm for quadratic assignment problems using Bender's decomposition. *European Journal Operation Research, 2*, 204-211.
[23] Kirkpatrick, S., Gelatt Jr, C., & Vecchi, M. (1983). Optimization by Simulated Annealing. *Science, 220*, 671-680.
[24] Koopmans, T., & Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica, 25*, 53-76.
[25] Krarup, J., & Pruzan, P. (1978). Computer-aided layout design. *Mathematical Programming Study, 9*, 75-94.
[26] Land, A. (1963). A problem of assignment with interrelated costs. *Operation Research Quarterly, 14*, 185-198.
[27] Lawler, E. (1963). The quadratic assignment problem. *Management Science, 9*, 586-599.
[28] Li, Y., Pardalos, P., & Resende, M. (1994). Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP. *Technical report, AT&T Bell Labratories*.
[29] Mills, P., Tsang, E., & Ford, J. (2003). Applying an extended guided local search to the quadratic assignment problem. *Annals of Operation Research, 118*, 121-135.
[30] Miranda, G., Luna, H., Mateus, G., & Ferreira, R. (2005). A performance guarantee heuristic for electronic components placement problems including thermal effects. *Computers and Operations Research, 32*, 2937-2957.
[31] Mirchandani, P., & Obata, T. (1979, May). Locational decisions with interactions between facilities: the quadratic assignment review. *Working paper Ps-79-1,Rensselaer Polytechnic Institute*.
[32] Misevicius, A. (2005). A tabu search algorithm for the quadratic assignment problem. *Computational Optimization and Applications, 30(1)*, 95-111.
[33] Nugent, C., Vollmann, T., & Ruml, J. (1969). An experimental comparison of techniques for the assignment of facilities to locations. *Journal of Operation Research, 16*, 150-173.
[34] Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman. *SIAM Review, 33*, 60-100.
[35] Pardalos, P., & Crouse, J. (1989). A parallel algorithm for the quadratic assignment problem. *In Proceeding of the Supercomputing 1989 Conference*, 351-369.
[36] Peng, T., Huanchen, W., & Dongme, Z. (1996). Simulated annealing for the quadratic assignment problem: A further study. *Computers and Industrial Engineering, 31(3-4)*, 925-928.
[37] Pollatschek, M., Gershoni, N., & Radday, Y. (1976). Optimization of the typewriter keyboard by simulation. *Angewandte Informatik, 17*, 438-439.
[38] Rabak, C., & Sichman, J. (2003). Using A-Teams to optimize automatic insertion of electronic components. *Advanced Engineering Informatics, 17 (2)*, 95-106.
[39] Roucairol, C. (1987). A parallel branch and bound algorithm for the quadratic assignment problem. *Discrete Applied Mathematcs, 18*, 211-225.
[40] Singh, S., & Sharma, P. (2006). Two-level modified simulated annealing based approach for

solving facility layout problem. *International Journal of Production Research. Department of Industrial and Management Engineering, Indian Institute of Technology Kanpur*.

[41] Steinberg, L. (1961). The backboard wiring problem: A placement algorithm. *SIAM Review, 3*, 37-50.

[42] Urban, T. (1998, 01). Solution procedures for daynamic facility layout problem. *Annals of operation research, 76*, 323-342.

[43] Wess, B., & Zeithofer, T. (2004). On the phase coupling problem between data memory layout generation and address pointer assignment. *Lecture Notes in Computer Science 3199*, 152-166.