JOURNAL OF APPLIED MATHEMATICS AND DECISION SCIENCES, 6(3), 143–153

A New Heuristic for the Quadratic Assignment Problem

ZVI DREZNER[†] zdrezner@fullerton.edu
College of Business and Economics, California State University, Fullerton, CA 92834.

Abstract. We propose a new heuristic for the solution of the quadratic assignment problem. The heuristic combines ideas from tabu search and genetic algorithms. Run times are very short compared with other heuristic procedures. The heuristic performed very well on a set of test problems.

Keywords: Quadratic Assignment, Heuristic Algorithms.

1. Introduction

The quadratic assignment problem is considered one of the most difficult optimization problems. It requires the location of n facilities on a given set of n sites, one facility at a site. A distance matrix $\{d_{ij}\}$ between sites and a cost matrix $\{c_{ij}\}$ between facilities are given. The cost f to be minimized over all possible permutations, calculated for an assignment of facility i to site $p(i)$ for $i = 1, \ldots, n$, is:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} d_{p(i)p(j)} \tag{1}$$

The first heuristic algorithm proposed for this problem was CRAFT [1]. More recent algorithms use metaheuristics such as tabu search [2], [14], [16], simulated annealing [4], [6], [19] and genetic algorithms [8]. For a complete discussion and list of references see [3], [17], [5].

2. The Algorithm

A starting solution is selected as the center solution. A "distance" Δp is defined for each solution (permutation p of the center solution). The distance Δp is the number of facilities in p that are not in their center solution site. The following are trivial properties of Δp:

[†] Requests for reprints should be sent to Zvi Drezner, College of Business and Economics, California State University, Fullerton, CA 92834.

1. For all single pair exchanges of the center solution $\Delta p = 2$.
2. There are no permutations with distance $\Delta p = 1$.
3. $\Delta p \leq n$ because no more than n facilities can be out of place.
4. If an additional pair exchange is applied on a permutation p, then $\Delta\Delta p$ (the change in Δp) is between -2 and +2. Note that only the additional pair of exchanged facilities affect the value of $\Delta\Delta p$. Therefore, $\Delta\Delta p$ is calculated in $O(1)$

We use this distance from the center solution as a tabu mechanism. We search the solution space starting from the center solution, and each step we search solutions with increasing distance from the center solution, until we reach a prespecified search depth $d \leq n$. We do not move back towards the center solution unless a solution better than the best found one is encountered. We perform all pair-wise exchanges of the center solution and keep the best K permutations, where K is a parameter of the algorithm. All these K permutations are different from the center solution in two sites and therefore their distance from the center solution is $\Delta p = 2$. Pair-wise exchanges of any of these K permutations lead to permutations which are different from the center one in either two or fewer sites, or three sites, or four sites. We generate two lists. One consists of the best K permutations with distance $\Delta p = 3$, and the other is a list of the best K permutations with $\Delta p = 4$. Next we check all pairwise exchanges of the best found K permutations with $\Delta p = 3$, updating possibly the list with $\Delta p = 4$, and creating a list of the best K permutations with distance $\Delta p = 5$, and so on, until we scan the best K permutations which are different from the original one in d sites. If throughout the process a solution better than the best found one is encountered, the center solution is replaced by the newly found best solution, and the process starts from the beginning. If no better solution is found throughout the scanning, the iteration is complete.

2.1. Description of One Iteration

A center solution is selected. Let p be a permutation of the center solution by successive pair exchanges. We keep three lists of solutions which we term $list_0$, $list_1$, and $list_2$. $list_0$ contains solutions with distance Δp. Similarly, $list_1$ and $list_2$ contain solutions whose distance is $\Delta p + 1$, and $\Delta p + 2$, respectively. The depth of the search is set to $d \leq n$. It is recommended to apply $d = n$ or very close to it.

1. A center solution is selected.

2. Start with $\Delta p = 0$. At the start, $list_0$ has one member (the center solution) and the other two lists are empty. The best found solution is the center solution.
3. Go over all the solutions in $list_0$. For each solution p:
   - evaluate all pair exchanges for that solution.
   - If the exchanged solution is better than the best found solution, update the best found solution and proceed to evaluate the rest of the exchanges. This is necessary in order to be able to apply the short cut (see the appendix).
   - If the distance of an exchanged solution is Δp or lower, ignore the permutation and proceed with scanning the exchanges of permutation p.
   - If its distance is $\Delta p + 1$ or $\Delta p + 2$, check it for inclusion in the appropriate list. This is performed as follows:
     - If the list is shorter than K or the solution is better than the worst list member, check if it is identical to a list member by first comparing its value of the objective function to those of list members, and only if it is the same as that of a list member the whole permutation is compared.
     - If an identical list member is found, ignore the permutation and proceed with scanning the exchanges of permutation p.
     - Otherwise,
       - If the list is of length K, the permutation replaces the worst list member,
       - If the list is shorter than K, the permutation is added to the list.
     - A new worst list member is identified.
4. If a new best found solution is found by scanning all the exchanges of permutation p, set the center solution to the new best found solution and go to Step 2.
5. Once all solutions in $list_0$ are exhausted, move $list_1$ to $list_0$, $list_2$ to $list_1$, and empty $list_2$.
6. Set $\Delta p = \Delta p + 1$.
7. If $\Delta p = d + 1$ stop the iteration.
8. Otherwise, return to Step 3.

The process of an iteration applies concepts from tabu search and genetic algorithms [10], [13]. In tabu search we disallow backward tracking thus forcing the search away from previous solutions. In our approach we proceed farther and farther away from the center solution because Δp increases throughout the iteration. As in tabu search we can go back *only* when going back leads to a solution better than the best found one. It resembles genetic algorithms because a "population of K members" is maintained during the iteration and populations of offspring, that replace adults in the population are generated. However, there is no merging of parents in our procedure. Our population is unisex as in primitive forms of life. Progression and improvement occur only due to mutations.

We constructed an algorithm that repeats the iterations several times. Once an iteration is completed, (i.e., the scanning of all lists did not yield a solution better than the best found one), we can start a new iteration. We can use either the best solution found throughout the last scan (which is not better than the previous center solution) as the new center solution, or the best solution in the last list (of distance $\Delta p = d$). The first option is similar to a steepest descent approach or an iteration of a tabu search with an empty tabu list. The second approach is similar to the diversification idea in tabu search, i.e. moving to a "different region" in the search space. We opted to combine the two and alternate between selecting the best one throughout the search, and the best one in the last list for the next iteration.

The Algorithm

1. Select a random center solution. It is also the best found solution.
2. Set a counter $c = 0$.
3. Select d randomly in $[n-4, n-2]$. Perform an iteration on the center solution.
4. If the iteration improved the best found solution go to step 2
5. Otherwise, advance the counter $c = c + 1$, and
    - If $c = 1, 3$ use the best solution in the list with $\Delta p = d$ as the new center solution and go to Step 3.
    - If $c = 2, 4$ use the best solution found throughout the scan (not including the old center solution) as the new center solution and go to Step 3.
    - If $c = 5$ stop.

3. Computational Experiments

The algorithm was coded in Microsoft PowerStation Fortran 4.0 and ran on a Portege Toshiba 600MHz Pentium III lap-top. We selected all symmetric problems, for which the optimal solution is not known, of up to 64 facilities. The data base on http://www.mim.du.dk/~sk/qaplib has all 19 problems. The problems are: Kra30a, Kra30b [11], Nug30 [12], Tho30, Tho40 [18], Esc32a-32h Esc64a [7], Ste36a-c [15], Sko42-64 [14], and Wil50 [19]. The problem name includes the number of facilities as part of it.

As will be seen from the results reported below, the new algorithm is very fast, and we believe it is much faster than other algorithms reported in the literature (problems with $n = 30$ facilities were solved 120 times with $K = 1$ in 3-4 seconds, i.e., 0.03 seconds per run. Problems with $n = 64$ facilities were solved 120 times with $K = 1$ in less than a minute, or less than half a second per run.) We also plan to use this algorithm as part of a genetic algorithm. These run times are the times required to generate an initial population of 120 members using $K = 1$. Preliminary results of such a genetic algorithm are very promising. Comparing and evaluating the advantage of our algorithm over other algorithms is very difficult. We therefore, compared the performance of the algorithm using different values of K. The results reported below are of about the same quality as those reported in [17] but the run times are much faster.

We ran the algorithm for $K = 1-6, 8, 10$. We ran each variant $\frac{120}{K}$ times and report the best solution found in these runs so that run times will be about the same. We chose "120" because it divides evenly the selected set of K's.

The results are summarized in Tables 1-2. In Table 3 we give the information about the instances in which the best known solution was not found for completeness of the presentation.

By examining tables 1-3 we conclude that

- Run times are very similar for various K's. Generally, they decrease with K.
- In most cases the best average solution is obtained for $K = 1$.
- The K for which the highest number of times that the best known solution is obtained was $K = 1$ in many cases. However, especially for larger problems, the number of times was higher for a larger K. Clear exceptions are:
    - Ste36c for which the highest number was recorded for $K = 4$ (41 cases for $K = 4$ compared with only 5 cases for $K = 1$);

Table 1. Comparison Between Different Population sizes ($K = 1 - 4$)

Problem	Best	$K = 1$			$K = 2$			$K = 3$			$K = 4$		
	Known	†	‡	*	†	‡	*	†	‡	*	†	‡	*
Kra30a	88900	62	0.63	3.4	70	0.58	3.2	60	0.69	3.1	53	0.81	3.0
Kra30b	91420	37	0.08	3.4	20	0.12	3.1	13	0.16	3.0	16	0.15	3.0
Nug30	6124	62	0.04	3.7	62	0.03	3.3	68	0.04	3.2	60	0.05	3.0
Tho30	149936	76	0.09	3.7	81	0.09	3.3	60	0.13	3.1	54	0.15	3.0
Esc32a	130	112	0.10	3.9	116	0.05	3.7	114	0.08	3.7	109	0.17	3.7
Esc32b	168	120	0	3.3	120	0	3.0	120	0	2.9	120	0	2.9
Esc32c	642	120	0	2.1	120	0	2.0	120	0	1.9	120	0	1.9
Esc32d	200	120	0	2.5	120	0	2.4	120	0	2.4	120	0	2.4
Esc32h	438	120	0	2.6	120	0	2.5	120	0	2.5	119	0.00	2.4
Ste36a	9526	8	0.49	7.2	5	0.64	6.7	5	0.63	6.5	14	0.59	6.4
Ste36b	15852	56	0.48	6.9	40	0.62	6.2	34	0.77	6.0	33	0.90	5.8
Ste36c	8239.11	5	0.25	7.2	2	0.34	6.6	4	0.31	6.4	41	0.28	6.1
Tho40	240516	4	0.19	10.1	1	0.23	9.2	1	0.22	8.9	2	0.23	8.7
Sko42	15812	63	0.06	12.6	58	0.05	11.4	36	0.07	11.2	35	0.09	10.7
Sko49	23386	7	0.13	21.8	2	0.14	20.0	6	0.14	19.6	3	0.14	18.9
Wil50	48816	3	0.08	22.9	0	0.08	21.1	0	0.09	19.9	1	0.09	19.6
Sko56	34458	0	0.19	34.1	0	0.20	31.2	0	0.21	30.0	0	0.21	29.8
Sko64	48498	0	0.19	54.7	0	0.20	50.4	1	0.22	49.3	2	0.23	48.4
Esc64a	116	120	0	18.5	120	0	17.5	120	0	17.2	120	0	17.2

† Number of times out of 120 that best known solution obtained
‡ Percentage of average solution over the best known solution
* Time in seconds per run (Each run consists of $\frac{120}{K}$ individual runs)

Table 2. Comparison Between Different Population sizes ($K = 5 - 10$)

Problem	Best	$K = 5$			$K = 6$			$K = 8$			$K = 10$		
	Known	†	‡	*	†	‡	*	†	‡	*	†	‡	*
Kra30a	88900	53	0.84	3.0	48	0.94	2.9	35	1.10	2.9	33	1.14	2.8
Kra30b	91420	18	0.17	2.9	18	0.20	2.9	9	0.26	2.8	12	0.26	2.8
Nug30	6124	39	0.07	3.0	32	0.11	2.9	29	0.08	2.8	17	0.13	2.8
Tho30	149936	72	0.11	2.9	50	0.17	2.9	52	0.18	2.8	49	0.20	2.7
Esc32a	130	108	0.15	3.7	110	0.14	3.8	104	0.22	3.7	102	0.28	3.8
Esc32b	168	120	0	2.9	120	0	2.9	120	0	2.9	120	0	2.9
Esc32c	642	120	0	1.9	120	0	1.9	120	0	1.9	120	0	1.9
Esc32d	200	120	0	2.3	120	0	2.3	119	0.01	2.3	120	0	2.3
Esc32h	438	119	0.00	2.4	120	0	2.4	113	0.03	2.4	108	0.05	2.4
Ste36a	9526	5	0.82	6.3	9	0.72	6.3	4	0.84	6.1	7	0.81	6.1
Ste36b	15852	23	1.04	5.6	31	1.03	5.6	18	1.45	5.4	20	1.51	5.4
Ste36c	8239.11	23	0.41	6.1	30	0.37	6.0	14	0.45	6.0	15	0.52	5.9
Tho40	240516	1	0.28	8.5	0	0.27	8.4	0	0.27	8.1	0	0.30	8.2
Sko42	15812	45	0.09	10.4	25	0.11	10.4	39	0.11	10.1	15	0.16	10.0
Sko49	23386	3	0.14	18.7	11	0.15	18.5	2	0.17	18.5	6	0.16	18.5
Wil50	48816	4	0.09	19.1	6	0.10	19.2	3	0.11	18.5	4	0.12	18.9
Sko56	34458	2	0.23	29.6	1	0.24	28.9	1	0.27	29.3	0	0.29	29.9
Sko64	48498	3	0.24	48.3	4	0.25	48.5	1	0.24	48.3	1	0.27	49.9
Esc64a	116	120	0	17.4	120	0	17.7	120	0	18.1	120	0	18.4

† Number of times out of 120 that best known solution obtained
‡ Percentage of average solution over the best known solution
* Time in seconds per run (Each run consists of $\frac{120}{K}$ individual runs)

Table 3. Instances Where the Best Known Solution was not Found

Problem	Best	First Instance				Second Instance				Third Instance			
	Known	K	Min	†	‡	K	Min	†	‡	K	Min	†	‡
Tho40	240516	6	240542	9	0.01	8	240542	7	0.01	10	240542	12	0.01
Wil50	48816	2	48824	6	0.02	3	48824	8	0.02				
Sko56	34458	1	34462	2	0.01	2	34462	1	0.01	3	34462	1	0.01
		4	34462	1	0.01	10	34462	1	0.01				
Sko64	48498	1	48502	4	0.01	2	48502	1	0.01				

† Number of times out of 120 that the minimum solution obtained
‡ Percentage of minimum solution over the best known solution

 – Sko49 achieved the highest number for $K = 6$;
 – Wil50 achieved the highest for $K = 6$,
 – Sko56 achieved the highest for $K = 5$; and
 – Sko64 achieved the highest for $K = 6$.

- For most problems, especially the larger ones, the average solution was within 0.2% of the best known solution. This is about the quality of the solutions for such problems obtained by slower algorithms [17].
- There were 12 instances (out of 152) in which the best known solution was not identified even once (see Table 3). However, in all 12 cases the best solution found was within 0.02% of the best known solution.

4. Conclusions

We proposed a new heuristic algorithm for the solution of the Quadratic Assignment problem. The algorithm combines features of tabu search and genetic algorithms. We plan to investigate the application of this algorithm as part of a genetic algorithm. Preliminary experiments with genetic algorithms incorporating the proposed algorithm are very promising.

The proposed algorithm is very fast compared with other heuristics. It is therefore difficult to compare it with other algorithms. However, we obtained solutions of similar quality to other algorithms in shorter run times. This demonstrates the superiority of the proposed algorithm over existing ones.

Appendix: The Short Cuts

The following short cuts are explained in [17]. Since we experimented only with symmetric problems, we present this short cut for symmetric problems with zero diagonal (i.e., the cost between a facility and itself, and the distance between identical locations is zero).

Let Δf_{rs} be the change in the cost f by exchanging the sites of facilities r and s. This is a similar concept to the derivative of f. There are $\frac{n(n-1)}{2}$ Δf_{rs}'s. It can be easily verified by Equation (1) that:

$$\begin{aligned} \Delta f_{rs} &= 2\sum_{i=1}^{n}\left\{c_{ir}[d_{p(i)p(s)} - d_{p(i)p(r)}] + c_{is}[d_{p(i)p(r)} - d_{p(i)p(s)}]\right. \qquad (2) \\ &= 2\sum_{i=1}^{n}\left\{[c_{ir} - c_{is}][d_{p(i)p(s)} - d_{p(i)p(r)}]\right\} \end{aligned}$$

Calculating Δf_{rs} by using Equation (2) requires only $O(n)$ time rather than $O(n^2)$ time required to calculate f by Equation (1).

Taillard [17] points to a faster formula to calculating Δf_{rs}. Define $\Delta_{uv} f_{rs}$ the change in the value of the objective function between the exchanged permutation by rs, and an additional exchanged pair uv. This is a similar concept to the second derivative of f. This change in the value of the objective function can be calculated in $O(1)$ (starting from the second iteration) if the pairs rs and uv are mutually exclusive. The formula is based on Δf_{uv} (the change in the value of the objective function from the *previous* permutation by exchanging the pair uv). Therefore, one needs to keep all the values of Δf_{ij} for all i, j, and K three times for a total of $3\frac{n(n-1)K}{2}$ values.

Since,

$$\Delta f_{uv} = 2\sum_{i=1}^{n}\left\{[c_{iu} - c_{iv}][d_{p(i)p(v)} - d_{p(i)p(u)}]\right\}$$

Then, by checking which terms change if an exchange uv is performed on a permutation where rs were exchanged:

$$\begin{aligned}\Delta_{uv} f_{rs} = \Delta f_{uv} &+ 2[c_{su} - c_{sv} - (c_{ru} - c_{rv})][d_{p(r)p(v)} - d_{p(r)p(u)}] \\ &+ 2[c_{ru} - c_{rv} - (c_{su} - c_{sv})][d_{p(s)p(v)} - d_{p(s)p(u)}]\end{aligned}$$

which leads to:

$$\begin{aligned}\Delta_{uv} f_{rs} = \Delta f_{uv} &+ 2[c_{su} + c_{rv} - c_{sv} - c_{ru}] \\ &[d_{p(s)p(u)} + d_{p(r)p(v)} - d_{p(s)p(v)} - d_{p(r)p(u)}] \qquad (4)\end{aligned}$$

Note that only $2n - 3$ pairs are not mutually exclusive and formula (2) can be used in these cases to evaluate $\Delta_{uv} f_{rs}$. Therefore, evaluating the change in the value of the objective function for all $\frac{n(n-1)}{2}$ possible pair exchanges (which is required for one step of the algorithm) requires $O(n^2)$ time. Therefore, one pass through all $O(n)$ distances from the center using the short cut requires $O(n^3 K)$ time.

References

1. Armour G.C., and E.S. Buffa (1963) "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities," *Management Science*, 9, 294-309.
2. Battiti R. and G. Tecchiolli (1994) "The Reactive Tabu Search," *ORSA Journal on Computing*, 6, 126-140.
3. Burkard R.E. (1990) "Locations with Spatial Interactions: The Quadratic Assignment Problem," In P.B. Mirchandani and R.L. Francis, editors, *Discrete Location Theory*, Wiley, Berlin.
4. Burkard R.E., and F. Rendl (1984) "A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems," *European Journal of Operations Research*, 17, 169-174.
5. Cela E. (1998) *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Academic Publishers, Dordrecht.
6. Connolly D.T. (1990) "An Improved Annealing Scheme for the QAP," *European Journal of Operational Research*, 46, 93-100.
7. Eschermann B., and H.J. Wunderlich (1990) "Optimized Synthesis of Self-Testable Finite State Machines," In 20th International Symposium on Fault-Tolerant Computing (FFTCS 20), Newcastle upon Tyne, 26-28th June, 1990.
8. Fleurent C., and J.A. Ferland (1994) "Genetic Hybrids for the Quadratic Assignment Problem," In P. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems*, volume 16, pages 173-187. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.
9. Giovannetti M. (1997) "Methaheuristic and exact algorithms for the quadratic assignment problem," University of Bologna, Cesena Site.
10. Glover F., and M. Laguna (1997) *Tabu Search*, Kluwer Academic Publishers.
11. Krarup J., and P.M. Pruzan (1978) "Computer-Aided Layout Design," *Mathematical Programming Study*, 9, 75-94.
12. Nugent C.E., T.E. Vollman, and J. Ruml (1968) "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations," *Operations Research*, 16, 150-173.

13. Said S. (1998) "Heuristic Search Methods," in G. Marcoulides (Ed.) *Modern Methods for Business Research*, Lawrence Erlbaum Associates, Mahwah, NJ.
14. Skorin-Kapov J. (1990) "Tabu Search Applied to the Quadratic Assignment Problem," *ORSA Journal on Computing*, 2, 33-45.
15. Steinberg L. (1961) "The Backboard Wiring Problem: a Placement Algorithm," *SIAM Review*, 3, 37-50.
16. Taillard E.D. (1991) "Robust Tabu Search for the Quadratic Assignment Problem," *Parallel Computing*, 17, 443-455.
17. Taillard E.D. (1995) "Comparison of Iterative Searches for the Quadratic Assignment Problem," *Location Science*, 3, 87-105.
18. Thonemann U.W., and A. Bolte. (1994) "An Improved Simulated Annealing algorithm for the Quadratic Assignment Problem," Working paper, School of Business, Department of Production and Operations Research, University of Paderborn, Germany.
19. Wilhelm M.R. and T.L. Ward (1987) "Solving Quadratic Assignment Problems by Simulated annealing," *IIE Transactions*, 19, 107-119.