

A New Genetic Algorithm for the Quadratic Assignment Problem

Zvi Drezner
College of Business and Economics
California State University-Fullerton
Fullerton, CA 92834.
and
Visiting Professor
Graduate School of Management
University of California-Irvine
Irvine, CA 92697-3125.

Abstract

In this paper we propose several variants of a new genetic algorithm for the solution of the quadratic assignment problem. We designed a special merging rule for creating an offspring that exploits the special structure of the problem. We also designed a new type of a tabu search which we term a concentric tabu search. This tabu search is applied on the offspring before consideration for inclusion in the population. The algorithm provided excellent results for a set of twenty nine test problems ranging between 30 and 100 facilities.

Key Words: *Quadratic Assignment, Heuristics, Genetic Algorithm, Memetic Algorithm, Tabu Search.*

1 Introduction

The quadratic assignment problem is considered one of the most difficult optimization problems to solve optimally. A rich literature exist for heuristic approaches for its solution. The problem is defined as follows:

A set of n possible sites are given and n facilities are to be located on these sites, one facility at a site. Let c_{ij} be the cost between facilities i and j and d_{ij} be the distance between sites i and j . The cost f to be minimized over all possible permutations, calculated for an assignment of facility

i to site $p(i)$ for $i = 1, \dots, n$, is:

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

The first heuristic algorithm proposed for this problem was CRAFT [2] which is a descent heuristic. More recent algorithms use metaheuristics such as tabu search [3, 20, 22], simulated annealing [5, 26], genetic algorithms [1, 10, 24], ant colonies search [11], or specially designed heuristics [7, 16]. For a complete discussion and list of references see [4, 6, 23].

In this paper we first describe the genetic algorithms in general, present the two merging processes used in the proposed genetic algorithms, and present three different procedures to be applied on offspring before consideration for inclusion in the population (which we term a post merging procedure (PMP)). Such algorithms are sometimes referred in the literature as “Memetic algorithms” [18]. In Section 3 we present extensive computational comparisons between all proposed variants. We summarize the results and propose future research in the conclusion section.

2 Genetic Algorithms

Genetic algorithms have proven to be quite successful for the solution of combinatorial problems. For a review see [14, 19]. Proposed genetic algorithms for the solution of the quadratic assignment problem are [1, 10, 24].

The framework of a genetic algorithm is:

1. randomly generate an initial population of solutions,
2. each generation, select pairs of population members and merge them to produce offspring,
3. select a population for the next generation from the existing population members and the offspring,

4. a stopping rule is used to stop the procedure.
5. The best solution found throughout the process is selected as the solution.

Mutations of population members or offspring are also considered. A mutation of a solution may constitute the application of a heuristic on the solution. One can use a single pair exchange or apply a post merging process (PMP) in lieu of a single pair exchange. One can apply a descent algorithm as a PMP. Other researchers [10, 23] suggest to use a short version of Robust-Tabu [22] on the offspring as a PMP.

2.1 The Merging Processes

The most crucial part of a successful genetic algorithm is the merging process of two parents to produce an offspring. For the process to be effective an offspring should be significantly better (in terms of its value of the objective function) than a randomly generated solution. Otherwise, we do not gain by the merging process. It is true that such an algorithm may find a good solution, but it does not have a significant advantage over repeating the PMP from randomly generated solutions the same number of times. Therefore, it is essential to find a merging rule which exploits the structure of the problem and is likely to use “good features” of the parents when creating an offspring.

We constructed merging processes which are similar to the successful merging procedure used in [8] for solving a network design problem. In [8] the network was divided into two cohesive parts by selecting a pivot node, assigning a count of “1” to all links directly connected to it, a count of “2” to all links connected to a link of count “1”, and so on. Each link gets a count. The median of these counts for all links is calculated. Links with a count below the median are taken from the first parent and links above the median are taken from the second parent. Links which have a count

Figure 1: The Distances

4	3	4	5	6
3	2	3	4	5
2	1	2	3	4
1	0	1	2	3
2	1	2	3	4

equal to the median are randomly assigned to one of the parents. This way each part of a parent is a connected part of the network. It is suggested in [8] to consider the n possible partitions (one for each pivot node) and select the best offspring of these n partitions for a post merging procedure and possible inclusion in the population.

For the quadratic assignment problem we suggest similar merging processes. We tried several variations of merging procedures and the following two were found to be the most effective. Both approaches are based on finding a median distance from a “pivot site”.

An illustration on a 5 by 5 problem with rectilinear distances is depicted in Figure 1. The pivot site has a distance of “0” and all other sites have positive distances. The median distance is “3”.

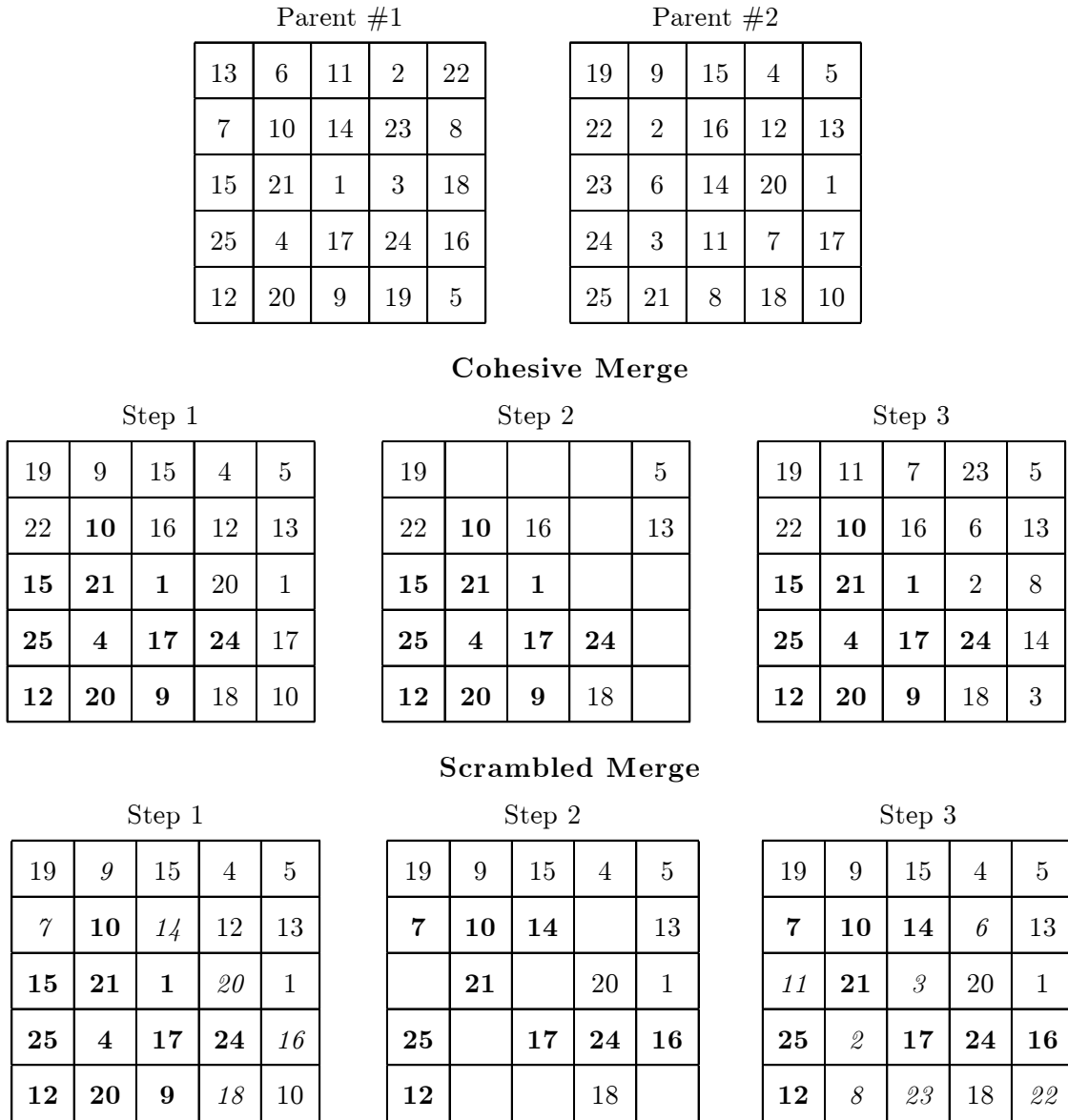
2.1.1 The Cohesive Merging Procedure

In the cohesive merging process we attempt to divide the sites into two cohesive parts and each has all its objects (facilities) from the same parent.

A pair of parents is randomly selected, and the parent with the better value of the objective function is selected as the first parent. If the two parents tie in the value of the objective function, one of them is arbitrarily selected as the first parent. The following is executed for every pivot site.

1. The median distance from the pivot site to all sites is calculated (this is done in the preamble and not every iteration).

Figure 2: The Cohesive and Scrambled Merges



2. A site which is closer than the median to the pivot site is assigned the facility from the first parent.
3. All other sites are assigned a facility from the second parent.
4. It is possible that some facilities are assigned twice and some are not assigned at all. Therefore,
 - (a) Go over all the facilities from left to right and create a list of unassigned facilities.
 - (b) Find all facilities that are assigned twice, and replace the site which is farther than the median (i.e., from the second parent) with a facility that is not assigned at all.
5. This completes the merge of the two selected parents for one pivot site.

The procedure is illustrated in Figure 2. In Step 1 we select the facilities from parent #1 for all sites with a distance below the median (distance of 3). These are denoted with boldface. The facilities for all other sites are taken from parent #2. In Step 2 we blank all facilities from parent #2 which also exist in the list of facilities taken from parent #1. Facilities 2, 3, 6, 7, 8, 11, 14, 23 are now missing and are entered in the blank cells to complete the merged offspring depicted in Step 3. The boldfaced facilities are taken from parent #1 and all the others from parent #2. Notice that the facilities taken from a parent form a cohesive set and do not intermingle among each other.

2.1.2 The Scrambled Merging Procedure

In the scrambled merging process we treat both parents equally and the parts taken from each parent may not form cohesive sets and facilities from both parents may intermingle with one another.

A pair of parents is randomly selected. The following is executed for every pivot site.

1. A site which is closer than the median to the pivot site is assigned the facility from the first parent.
2. A site which is farther than the median is assigned the facility from the second parent.
3. Sites which are exactly at the median distance are assigned one of the two facilities (from either the first or the second parent) at random.
4. Once this first phase is completed, there might be facilities which are assigned to two sites and an equal number of facilities which are assigned to no site.
5. As a second phase we scan all facilities which are assigned to two sites and replace one of the facilities with a facility which is assigned to no site.
6. Once this phase is completed, we get a feasible assignment.

The procedure is illustrated in Figure 2. In Step 1 we select facilities from parent #1 for all sites with a distance below the median (denoted with boldface). The facilities for all sites with a distance greater than the median are taken from parent #2 and the facilities for the six sites with a distance equal to 3 are randomly taken from one of the parents and are denoted in italics. One of the facilities that appear twice in Step 1 is randomly blanked resulting in Step 2. The facilities originated from Parent #1 are denoted in boldface and the others taken from parent #2 in regular font. Facilities 2, 3, 6, 8, 11, 22, 23 do not appear in Step 2. They are randomly entered into the blank cells resulting in the merged offspring depicted in step 3. Note that facilities from different parents intermingle with one another in the final scheme.

2.1.3 Comments

Quadratic assignment problems have a set of weights and a set of distances. The problem remains the same if we exchange the roles of weights and distances. However, the suggested merging procedures are most effective when we use the original distance matrix as distances. This is because some problems may have many zero weights. For such problems, if weights are used as distances, the median distance for some pivot sites may be zero and the merging procedures may not be effective.

A merging process is repeated n times, once for every site being the “pivot site”, and the offspring with the best value of the objective function is selected for PMP and possible inclusion in the population. Note that the merging process requires negligible run time compared with the run time required for the PMP. Therefore, generating n merges and selecting the best for a PMP rather than generating only one merge does not add noticeable run time to the process.

2.2 The Post Merging Procedures

An efficient PMP can improve the resulting genetic algorithm. For our PMP we opted to experiment with a descent heuristic, a simple tabu search, and a new tabu search which we term “a concentric tabu search”. These variants of the PMP are summarized below.

2.2.1 The Descent Heuristic

For completeness we describe the descent heuristic:

1. Select a starting solution.
2. Check the change in the value of the objective function for all pair-wise exchanges of facilities.
3. If an improving exchange is found, the best improving exchange is executed and go to Step 2.

4. If no improving exchange is found, the algorithm terminates.

2.2.2 The Simple Tabu Search

Tabu search techniques can be very complicated. See [13] for a complete description of these techniques. We aimed for a simple tabu search which requires about double the time required for the descent heuristic. We therefore start with the descent heuristic (as all tabu searches do), and once it terminates, we continue with the tabu search for at most twice the number of iterations that were performed by the descent heuristic (but at least 50 iterations). For such a short tabu search the number of iterations is not sufficient to exploit elaborate tabu rules. We therefore constructed a simple tabu rule. The tabu list consists of the individual members of the exchanged pair (i.e., two facilities are added to the tabu list each iteration) and we keep facilities in the tabu list for ten iterations. We empty the tabu list whenever a new best known solution is found. This is summarized as follows:

1. Select a starting solution and perform the descent heuristic. The terminal solution of the descent heuristic is defined as the current solution and the best known solution. The number of iterations of the descent heuristic is h . Empty the tabu list.
2. Repeat the following $\max\{2h, 50\}$ times:
 - (a) Check all pair-wise exchanges of facilities in the current solution.
 - (b) If a solution better than the best known solution is found, the best improving exchange is performed, the tabu list is emptied, and the next iteration starts.
 - (c) If no exchanged solution is better than the best known solution, the best exchange (whether improving or not) between two facilities, both not in the tabu list, is performed. The two exchanged facilities are added to the tabu list. If the list exceeds 20 facilities,

the two facilities with the largest tenure in the tabu list are removed from the tabu list.

The next iteration starts.

2.2.3 The Concentric Tabu Search

The concentric Tabu Search is very similar to the algorithm described in [7] using a population of one. The best known solution serves as the “center” of the search. A “distance”, Δp , of a solution is defined as the number of facilities which are located at different sites than the “center” solution. Note that $0 \leq \Delta p \leq n$ and $\Delta p \neq 1$. The search proceeds such that every iteration the distance from the center increases, thus forbidding moves back “towards” the center. The depth of the search $d \leq n$ is the maximum distance that the search will attempt. This number should be close to n . We opted to randomly generate d in $[n - 4, n - 2]$.

The Concentric Tabu Algorithm

1. Set a counter to 1 and select a random solution as a starting solution. Go to Step 3
2.
 - if the counter is equal to 2 or 4 select the best solution with $\Delta p = n$ as a starting solution.
 - if the counter is equal to 3 select the best solution found for $2 \leq \Delta p \leq n$ as the starting solution.
 - if the counter is equal to 5, stop the algorithm with the best known solution as its result.
3. The starting solution is defined as the center solution, and the current solution. Set $\Delta p = 0$.
Randomly select the depth of the search d in $[n - 2, n - 4]$.
4. Evaluate all pair exchanges of the current solution. For each pair exchange do:

- (a) If the exchanged solution is better than the best known solution, update the best known solution.
 - (b) If the order of the exchanged solution is Δp or lower, ignore it.
 - (c) If its order is either $\Delta p + 1$ or $\Delta p + 2$, compare it with the best found solution for that order and update the best found solution for that order if necessary.
5. If a solution better than the best known solution was found, select the new best known solution as a new starting solution, set the counter to 1 and go to Step 3
 6. Move the best found solution for $\Delta p + 1$ to the current solution. The best found solution for $\Delta p + 2$ is now the best found solution for $\Delta p + 1$.
 7. Set $\Delta p = \Delta p + 1$.
 8. If $\Delta p < d + 1$ go to Step 4.
 9. Otherwise, advance the counter by 1 and go back to Step 2

The concentric tabu algorithm uses a “distance” as a tabu mechanism, and borrows the diversification idea by diversifying to another region of the solution space if the procedure fails to find a better best known solution for all “distances”. We alternate by selecting either the best solution with $\Delta p = d$ which is “far” from the center, or the best solution found throughout the search for the start of the next iteration.

Note that the short cut explained below for calculating the change in the objective function when the exchanged pair is mutually exclusive of the previously exchanged pair [23], can be implemented for the descent, and the two types of the tabu search.

2.2.4 The Short Cut

Since we experimented only with symmetric problems with zero diagonals for weights and distances, we present a simplified version of the short cut for this case. Let Δf_{rs} be the change in the objective function when facilities r and s are exchanged. It can be easily verified by Equation (1) that:

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

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 [23] points to a faster formula. 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 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 ij , a total of $\frac{n(n-1)}{2}$ values.

Since,

$$\Delta f_{uv} = 2 \sum_{i=1}^n \left\{ [c_{iu} - c_{iv}] [d_{p(i)p(v)} - d_{p(i)p(u)}] \right\}\quad (3)$$

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

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

$$+ 2[c_{ru} - c_{rv} - (c_{su} - c_{sv})][d_{p(s)p(v)} - d_{p(s)p(u)}] \quad (4)$$

which leads to:

$$\Delta_{uvf_{rs}} = \Delta f_{uv} + 2[c_{su} - c_{sv} - c_{ru} + c_{rv}][d_{p(r)p(v)} - d_{p(r)p(u)} - d_{p(s)p(v)} + d_{p(s)p(u)}] \quad (5)$$

Note that only $2n - 3$ pairs are not mutually exclusive and formula (2) can be used in these cases to evaluate $\Delta_{uvf_{rs}}$. 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 iteration of the descent or tabu algorithm) requires $O(n^2)$ time.

2.3 The Specific Genetic Algorithm

We constructed a simple genetic algorithm based on the merging procedures and the post merging procedures described above. The population size is P and the number of generations is G .

1. P solutions are randomly selected and a PMP is applied on each creating the starting population.
2. Two population members are randomly selected to be parents. The quality of a population member does not affect the probability it is selected.
3. The two parents are merged n times, once for each “pivot site”, and the best merged permutation selected.
4. A PMP is applied on the best merged permutation creating an offspring.
5. If the offspring is better than the worst population member, then it is compared with all existing population members. Only population members with the same value of the objective function need to be compared.

- If it is not identical to an existing population member, the offspring replaces the worst population member.
 - If it is identical to an existing population member, the offspring is ignored.
6. Repeat Steps 2-5 G times.
 7. The best population member after G generations is selected as the solution.

We noticed that other genetic algorithms do not check whether an offspring is identical to a population member prior to including it in the population. If a population has identical members, two identical parents produce an identical offspring, regardless of the merging rule, and the impact of the PMP is just applying additional iterations on identical members. This may also cause early convergence to an inferior solution. It is possible that researchers believed that offspring which are identical to an existing population member are very rare. We have found in our experiments that many offspring were rejected for inclusion in the population because they were identical to a population member. It is actually quite common to obtain an offspring which is identical to a population member.

3 Computational Experiments

The algorithms were coded in Microsoft PowerStation Fortran 4.0 and run on a lap-top Toshiba Portege 7200 600MHz Pentium III computer. We tested the algorithms on twenty nine problems. The problems are given in the data base on <http://www.mim.du.dk/~sk/qaplib>. We selected all symmetric problems with $30 \leq n \leq 100$ for which the optimal solution is not known. The problems are: Kra30a, Kra30b [15], Nug30 [17], Tho30, Tho40 [25], Esc32a-32h, Esc64a (total of 6 problems) [9], Ste36a-c (3 problems) [21], Sko42-Sko100a-f (a total of 13 problems) [20] and Wil50, Wil100

[26]. The number of facilities is part of the problem name.

After many experiments with moderately sized problems ($30 \leq n \leq 64$) we selected a population size of 100. The number of generations for the concentric tabu was set to $\max\{20n, 1000\}$. The number of generations for the descent and the simple tabu was set to double these values. We noticed an improvement in the results of the algorithms when population size is increased (and number of generations increased proportionally). However, in order to stay within reasonable run times, we opted to experiment with a fixed population size of 100.

3.1 Comparing Different Merging Procedures

In order to determine the best merging procedure we tested all the variants on the 22 moderately sized problems ($30 \leq n \leq 90$). We tested the cohesive merging procedure, the scrambled merging procedure, TS/FF suggested by Tate and Smith [24] and Fleurent and Ferland [10] which was used in most other genetic algorithms, and for comparison no genetic procedure, that is, repeating the PMP the same number of times as the genetic algorithms ($\max\{20n, 1000\} + 100$) from randomly generated starting solutions. In Table 1 we report for each of the 22 problems and each merging procedure, the number of times the best known solution was obtained, the percent average of the solutions over the best known solution, and the run time in minutes per run.

Some observations from Table 1:

- Run times for the concentric tabu are very short. As can be easily calculated from the run times given for No Genetic, the concentric tabu search requires about 0.025 seconds for $n = 30$ problems. Run times increases to 1.32 seconds for $n = 90$ problems suggesting a run time of $O(n^{3.6})$.
- No Genetic has the longest run times. Run time seem to decrease a bit for TS/FF and

Table 1: Comparison Between Different Merging Procedures

Problem	Best Known	No Genetic			TS/FF			Cohesive			Scrambled		
		†	‡	*	†	‡	*	†	‡	*	†	‡	*
Kra30a	88900	20	0	0.45	20	0	0.43	20	0	0.33	20	0	0.32
Kra30b	91420	20	0	0.44	20	0	0.43	20	0	0.33	20	0	0.31
Nug30	6124	20	0	0.49	20	0	0.47	20	0	0.37	20	0	0.33
Tho30	149936	20	0	0.49	20	0	0.46	20	0	0.35	20	0	0.33
Esc32a	130	20	0	0.52	20	0	0.51	20	0	0.35	20	0	0.37
Esc32b	168	20	0	0.43	20	0	0.42	20	0	0.30	20	0	0.30
Esc32c	642	20	0	0.29	20	0	0.28	20	0	0.27	20	0	0.27
Esc32d	200	20	0	0.34	20	0	0.33	20	0	0.28	20	0	0.28
Esc32h	438	20	0	0.34	20	0	0.34	20	0	0.29	20	0	0.29
Ste36a	9526	6	0.114	0.95	8	0.063	0.91	19	0.005	0.55	16	0.021	0.65
Ste36b	15852	20	0	0.91	20	0	0.86	20	0	0.61	20	0	0.68
Ste36c	8239.11	1	0.107	0.95	14	0.024	0.89	14	0.039	0.59	18	0.010	0.66
Tho40	240516	3	0.034	1.34	3	0.025	1.32	5	0.010	0.98	5	0.015	0.91
Sko42	15812	20	0	1.66	20	0	1.60	20	0	1.15	20	0	1.20
Sko49	23386	10	0.032	2.89	14	0.022	2.81	17	0.009	2.13	18	0.007	2.16
Wil50	48816	3	0.024	3.08	7	0.014	2.94	18	0.002	1.99	18	0.002	1.85
Sko56	34458	0	0.041	5.01	0	0.046	4.83	19	0.001	3.24	17	0.002	3.29
Sko64	48498	3	0.043	9.09	1	0.035	8.96	20	0	5.85	19	0.000	6.01
Esc64a	116	20	0	3.21	20	0	3.19	20	0	3.05	20	0	3.10
Sko72	66256	0	0.120	15.76	0	0.115	15.50	10	0.014	8.36	7	0.013	7.74
Sko81	90998	0	0.124	25.52	0	0.112	25.43	5	0.014	13.30	4	0.019	12.78
Sko90	115534	0	0.139	41.81	0	0.126	41.63	4	0.011	22.35	2	0.019	19.52

† Number of times out of 20 that best known solution obtained

‡ Percentage of average solution over the best known solution

* Time in minutes per run

are significantly lower for Cohesive and Scrambled. We noticed, especially for Cohesive and Scrambled, that as the generations progress run times for each generation decreases significantly. This means that fewer PMP iterations are required per offspring as the generations progress because the offspring are generally better than a random solution. This phenomenon is hardly observed for TS/FF and, of course, does not occur in No Genetic. This indicates that the offspring generated by TS/FF are not much better than random solutions, but the offspring generated by Cohesive and Scrambled are.

- The widely used TS/FF merging procedure performs only slightly better than just No Genetic. Run times for TS/FF are slightly shorter.
- Both the Cohesive and the Scrambled merging procedures are significantly better than TS/FF and also run faster.
- One can view the difference between the No Genetic results and the three genetic procedures results as the “contribution” of the genetic component to the procedures.
- There seem to be a few anomalies in Table 1. Problems Ste36c and Tho40 seem to yield somewhat poorer results than their size would suggest. TS/FF even performed better than the Cohesive procedure on Ste36c. Another anomaly that we did not further investigate is that No Genetic performed better than TS/FF on problems Sko56 and Sko64. We ran Ste36c and Tho40 100 times to get more reliable comparisons. The results were:

Merging Procedure	Ste36c		Tho40	
	Times obtained out of 100	Percentage over best	Times obtained out of 100	Percentage over best
No Genetic	14	0.094	12	0.033
TS/FF	51	0.041	13	0.026
Cohesive	68	0.038	27	0.010
Scrambled	88	0.011	24	0.016

These results confirm the relative performance of the Cohesive and Scrambled procedures for Ste36c, and improved a bit the superiority of Cohesive over Scrambled for Tho40. The better performance of TS/FF over Cohesive for Ste36c in Table 1 was not confirmed and can be attributed to a statistical oddity.

- The Cohesive and Scrambled procedures perform almost equally well. With one exception, (problem Ste36c) the cohesive procedure performed almost equally well or slightly better than the scrambled procedure. More experiments are needed to establish the superiority of one procedure over the other. The relative performance of these two procedures may depend on the problem solved. Since the Cohesive merging procedure seems to perform slightly better, especially for larger problems, it was selected as the merging procedure for further experiments.
- One may wonder whether it is just a matter of luck that the nine smallest problems all found the best known solution in all 20 runs. To further investigate this issue, we ran the Nug30 problem 100,000 times using the concentric tabu search. The optimal solution of 6124 was found 532 times, or 0.532% of the time. The probability that the optimal solution is found at least once in 1100 runs (as in the column No Genetic in Table 1) is $1 - (1 - 0.00532)^{1100} = 0.99717$. The probability that all 20 runs of no genetic find the optimal solution is $0.99717^{20} = 0.945$. Hence, the probability of getting 20 out of 20 for the Nug30 problem is 94.5%. Since the genetic algorithms do generally better than No Genetic, this probability is a lower bound for the other three columns.

3.2 Comparing the Three Post Merging Procedures

For these comparisons we used the cohesive merging procedure and compared the descent, simple tabu, and concentric tabu as PMPs. In order to consume about the same computer time for each variant, the problems were solved 200 times, 100 times, and 20 times for the descent, simple tabu, and concentric tabu, respectively. The results are summarized in Table 2.

Examining Table 2 leads to the following conclusions

- Run times are very reasonable for each variant. For $n = 100$ problems run time is about 3.4, 5, and 34 minutes for descent, simple tabu, and concentric tabu, respectively.
- The concentric tabu variant found the best known solution at least once for all 29 problems; The simple tabu variant found the best known solution at least once for 27 out of 29 problems; the descent variant found the best known solution at least once in 26 out of 29 problems.
- The concentric tabu variant found the best known solution in all 20 runs for 13 out of the 29 problems.
- The average solution of the concentric tabu variant never exceeded 0.05% over the best known solution for any of the 29 problems. The average for all 29 problems was only 0.008% over the best known solution. The average for the $n = 100$ problems was only 0.017% over the best known solution.
- For the simple tabu variant, the average solution for all 29 problems was 0.037% over the best known solution, and the average for $n = 100$ problems was 0.060% over the best known solution.
- The descent variant performed quite well for most problems. However, for three problems the average solution was over 0.2% over the best known solution. The average solution for all 29

Table 2: Comparison Between Genetic Algorithms Using Different PMPs

Problem	Best Known	Descent			Simple Tabu			Concentric Tabu		
		†	‡	Time*	†	‡	Time*	†	‡	Time*
# of runs:		200			100			20		
Kra30a	88900	162	0.253	0.06	93	0.089	0.09	20	0	0.33
Kra30b	91420	124	0.037	0.06	79	0.019	0.09	20	0	0.33
Nug30	6124	160	0.013	0.06	99	0.001	0.10	20	0	0.37
Tho30	149936	192	0.009	0.06	100	0	0.10	20	0	0.35
Esc32a	130	144	0.569	0.06	100	0	0.07	20	0	0.35
Esc32b	168	200	0	0.05	100	0	0.08	20	0	0.30
Esc32c	642	200	0	0.05	100	0	0.06	20	0	0.27
Esc32d	200	200	0	0.05	100	0	0.06	20	0	0.28
Esc32h	438	200	0	0.05	100	0	0.06	20	0	0.29
Ste36a	9526	49	0.246	0.08	37	0.149	0.12	19	0.005	0.55
Ste36b	15852	195	0.015	0.08	100	0	0.14	20	0	0.61
Ste36c	8239.11	73	0.142	0.08	59	0.066	0.12	14	0.039	0.59
Tho40	240516	4	0.069	0.13	4	0.042	0.23	5	0.010	0.98
Sko42	15812	173	0.014	0.16	96	0.001	0.30	20	0	1.15
Sko49	23386	2	0.107	0.28	12	0.062	0.48	17	0.009	2.13
Wil50	48816	26	0.038	0.25	42	0.011	0.47	18	0.002	1.99
Sko56	34458	63	0.054	0.42	59	0.007	0.72	19	0.001	3.24
Sko64	48498	69	0.051	0.73	65	0.019	1.23	20	0	5.85
Esc64a	116	200	0	0.40	100	0	0.49	20	0	3.05
Sko72	66256	1	0.112	0.93	9	0.056	1.45	10	0.014	8.36
Sko81	90998	0	0.087	1.44	0	0.058	2.18	5	0.014	13.30
Sko90	115534	3	0.139	2.31	4	0.073	3.51	4	0.011	22.35
Sko100a	152002	7	0.114	3.42	3	0.070	5.11	5	0.018	33.55
Sko100b	153890	6	0.096	3.47	17	0.042	5.11	10	0.011	34.05
Sko100c	147862	2	0.075	3.22	11	0.045	4.69	5	0.003	33.80
Sko100d	149576	0	0.137	3.45	0	0.084	5.15	1	0.049	33.90
Sko100e	149150	4	0.071	3.31	17	0.028	4.70	18	0.002	30.67
Sko100f	149036	1	0.148	3.55	1	0.110	5.25	1	0.032	35.74
Wil100	273038	0	0.076	3.51	3	0.043	5.24	5	0.002	33.11

† Number of times out of the corresponding number of runs
that best known solution obtained

‡ Percentage of average solution over the best known solution

* Time in minutes per run

problems was 0.092% over the best known solution, and for $n = 100$ problems it was 0.102% over the best known solution.

- It is clear that the small increase in run time for the simple tabu over the descent PMP is justified. Therefore, using the descent PMP is not recommended. The concentric tabu definitely performs the best (only 0.008% over the best known solution, on the average), but it is not clear whether the increased run time over the simple tabu is justified. Getting a solution within an average of 0.037% over the best known solution, by using the simple tabu PMP and requiring only 5 minutes of computer time for $n = 100$ problems, is probably acceptable for most situations.

3.3 Comparison with [1]

We compared the results for our problems with those reported in [1]. Two of the problems are not reported in [1] and thus we compare the results for 27 problems. The results are given in Table 3. Run times in [1] are reported for GA-1 and for GA-3 they state that run times are about double those of GA-1. We therefore report double the times of GA-1. Note that Ahuja et al. [1] used an HP-6000 platform which is much faster than our lap-top. Therefore, any run times comparison is heavily skewed in our favor.

1. We found the best known solution *in all 20 runs* for 12 problems out of 27. Ahuja et al. [1] found the best known solution once (they ran each problem only once) for 9 out of 27 problems.
2. We found the best known solution at least once out of 20 runs for all 27 problems.
3. Our average percentage over the best known solution was 0.007% for these 27 problems compared with an average of 0.137% for theirs!

Table 3: Comparison Between The Results in this Paper and [1]

Problem	Best Known	Concentric Tabu			Ahuja et al. (2000)[1]	
		†	‡	Time*	‡	Time*
Kra30a	88900	20	0	0.33	0	5.02
Kra30b	91420	20	0	0.33	0	5.51
Nug30	6124	20	0	0.37	0.070	5.90
Tho30	149936	20	0	0.35	0	6.59
Esc32a	130	20	0	0.35	0	6.36
Esc32b	168	20	0	0.30	0	6.67
Esc32c	642	20	0	0.27	0	6.49
Esc32d	200	20	0	0.28	0	5.88
Esc32h	438	20	0	0.29	0	5.82
Ste36a	9526	19	0.005	0.55	0.270	11.83
Ste36b	15852	20	0	0.61	**	**
Ste36c	8239.11	14	0.039	0.59	**	**
Tho40	240516	5	0.010	0.98	0.320	15.97
Sko42	15812	20	0	1.15	0.250	16.77
Sko49	23386	17	0.009	2.13	0.210	20.87
Wil50	48816	18	0.002	1.99	0.070	35.25
Sko56	34458	19	0.001	3.24	0.020	49.60
Sko64	48498	20	0	5.85	0.220	63.14
Esc64a	116	20	0	3.05	0	43.85
Sko72	66256	10	0.014	8.36	0.290	84.63
Sko81	90998	5	0.014	13.30	0.200	182.74
Sko90	115534	4	0.011	22.35	0.270	211.63
Sko100a	152002	5	0.018	33.55	0.210	276.80
Sko100b	153890	10	0.011	34.05	0.140	245.49
Sko100c	147862	5	0.003	33.80	0.200	338.57
Sko100d	149576	1	0.049	33.90	0.170	338.37
Sko100e	149150	18	0.002	30.67	0.240	352.12
Sko100f	149036	1	0.032	35.74	0.290	357.98
Wil100	273038	5	0.002	33.11	0.200	342.40

† Number of times out of the corresponding number of runs that best known solution obtained

‡ Percentage of average solution over the best known solution

* Time in minutes per run

** Not reported

4. Their run times are about 10-20 times longer and were performed on a faster platform.
5. In conclusion, our method was about 20 times better both on the quality of the solution and run time.

4 Conclusions

A new genetic algorithm for the solution of the quadratic assignment problem is proposed. The merging rule for parents, which exploits the special structure of the problem, is shown to be superior to other merging rules which are reported in the literature. The proposed genetic procedure obtained excellent solutions for a set of test problems. The average performance of this algorithm is much better than the average performance of other algorithms reported in the literature.

The concentric tabu definitely performs the best (only 0.008% over the best known solution, on the average), but it is not clear whether the increased run time over the simple tabu is justified. Getting a solution within an average of 0.037% over the best known solution (by using the simple tabu post merging procedure and requiring only 5 minutes of computer time for problems with 100 facilities) is probably acceptable for most situations.

For future research we propose to examine the new merging rules with other genetic procedures and other post merging procedures. It may be possible to improve the performance of the proposed genetic procedure by changing the way generations are updated and parents selected. Other possible post merging procedures such as robust tabu [22], or more iterations on the simple tabu search employed in this paper, may also improve the performance of the genetic algorithm.

References

- [1] Ahuja R.K., J.B. Orlin, and A. Tiwari (2000) "A Descent Genetic Algorithm for the Quadratic Assignment Problem," *Computers and Operations Research*, 27, 917-934.

- [2] Armour G.C., and E.S. Buffa (1963) "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities," *Management Science*, 9, 294-309.
- [3] Battiti R. and G. Tecchiolli (1994) "The Reactive Tabu Search," *ORSA Journal on Computing*, 6, 126-140.
- [4] 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.
- [5] Burkard R.E., and F. Rendl (1984) "A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems," *European Journal of Operations Research*, 17, 169-174.
- [6] Cela E. (1998) *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Academic Publishers, Dordrecht.
- [7] Drezner Z. (2001) "Heuristic Algorithms for the Solution of the Quadratic Assignment Problem," *Journal of Applied Mathematics and Decision Sciences*, to appear.
- [8] Drezner Z. and S. Salhi (2001) "Using Metaheuristics for the One-Way and Two-Way Network Design Problem," *Naval Research Logistics*, to appear.
- [9] 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.
- [10] 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.
- [11] Gambardella L., E. Taillard, and M. Dorigo (1999) "Ant Colonies for the Quadratic Assignment Problem," *Journal of the Operational Research Society*, 50, 167-176.
- [12] Giovannetti M. (1997) "Metaheuristic and exact algorithms for the quadratic assignment problem," University of Bologna, Cesena Site.
- [13] Glover F., and M. Laguna (1997) *Tabu Search*, Kluwer Academic Publishers.
- [14] Goldberg D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Wokingham, England.
- [15] Krarup J., and P.M. Pruzan (1978) "Computer-Aided Layout Design," *Mathematical Programming Study*, 9, 75-94.
- [16] Li, Y., P.M. Pardalos, and M. Resende (1994) "A Descent Randomized Adaptive Search Procedure for the Quadratic Assignment Problem," in P. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems*, volume 16, pages 237-261. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.
- [17] 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.

- [18] Radcliffe N. J. (1994) "Formal Memetic Algorithms," in *Evolutionary Computing*, T. Fogarty (Ed.), Springer Lecture Notes in Computer Science, 865, 250-263.
- [19] Said S. (1998) "Heuristic Search Methods," in G. Marcoulides (Ed.) *Modern Methods for Business Research*, Lawrence Erlbaum Associates, Mahwah, NJ.
- [20] Skorin-Kapov J. (1990) "Tabu Search Applied to the Quadratic Assignment Problem," *ORSA Journal on Computing*, 2, 33-45.
- [21] Steinberg L. (1961) "The Backboard Wiring Problem: a Placement Algorithm," *SIAM Review*, 3, 37-50.
- [22] Taillard E.D. (1991) "Robust Tabu Search for the Quadratic Assignment Problem," *Parallel Computing*, 17, 443-455.
- [23] Taillard E.D. (1995) "Comparison of Iterative Searches for the Quadratic Assignment Problem," *Location Science*, 3, 87-105.
- [24] Tate D.M. and A.E. Smith (1995) "A Genetic Approach to the Quadratic Assignment Problem," *Computers and Operations Research*, 22, 73-83.
- [25] 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.
- [26] Wilhelm M.R. and T.L. Ward (1987) "Solving Quadratic Assignment Problems by Simulated annealing," *IIE Transactions*, 19, 107-119.