Chapter 10
Metaheuristic Algorithms for the Quadratic Assignment Problem: Performance and Comparison

Andreas Beham, Michael Affenzeller, and Erik Pitzer

Abstract. The quadratic assignment problem is among the harder combinatorial optimization problems in that even small instances might be difficult to solve and for different algorithms different instances pose challenges to solve. Research on the quadratic assignment problem has thus focused on developing methods that defy the problem's variety and that can solve a vast number of instances effectively. The topic of this work is to compare the performance of well-known "standard" metaheuristics with specialized and adapted metaheuristics and analyze their behavior. Empirical validation of the results is performed on a highly diverse set of instances that are collected and published in form of the quadratic assignment problem library. The data in these instances come from real-world applications on the one hand and from randomly generated sources on the other hand.

10.1 Introduction

10.1.1 Quadratic Assignment Problem

The quadratic assignment problem (QAP) was introduced in [15] and is a well-known problem in the field of operations research. It is the topic of many studies, treating the improvement of optimization methods as well as reporting successful application to practical problems in keyboard design, facility layout planning and re-planning as well as in circuit design [10, 13, 7]. The problem is NP hard in general and, thus, the best solution cannot easily be computed in polynomial time. Many

Andreas Beham · Michael Affenzeller · Erik Pitzer
Heuristic and Evolutionary Algorithms Laboratory
School of Informatics, Communication and Media
University of Applied Sciences Upper Austria, Research Center Hagenberg
Softwarepark 11, 4232 Hagenberg, Austria
e-mail: {andreas.beham,michael.affenzeller,
erik.pitzer}@fh-hagenberg.at

R. Klempous and J. Nikodem (eds.), *Innovative Technologies in Management and Science*,
Topics in Intelligent Engineering and Informatics 10, DOI: 10.1007/978-3-319-12652-4_10

different optimization methods have been applied, among them popular metaheuristics such as tabu search [20, 14] and genetic algorithms [8].

The problem can be described as finding the best assignment for a set of facilities to a set of locations so that each facility is assigned to exactly one location which in turn houses only this facility. An assignment is considered better than another when the flows between the assigned facilities have to be hauled over smaller distances.

The QAP is also a generalization of the traveling salesman problem (TSP). Conversion of a TSP can be achieved by using a special weight matrix [15, 17] where the flow between the "facilities" is modeled as a ring that involves all of them exactly once. The flow in this case can be interpreted as the salesman that travels from one city to another.

More formally the problem can be described by an $N \times N$ matrix W with elements w_{ik} denoting the weights between facilities i and k and an $N \times N$ matrix D with elements d_{xy} denoting the distances between locations x and y. The goal is to find a permutation π with $\pi(i)$ denoting the element at position i so that the following objective is achieved:

$$\min \sum_{i=1}^{N} \sum_{k=1}^{N} w_{ik} \cdot d_{\pi(i)\pi(k)} \tag{10.1}$$

The complexity of evaluating the quality of an assignment according to Eq. (10.1) is $O(N^2)$, however several optimization algorithms move from one solution to another through small changes, such as by swapping two elements in the permutation. These moves allow to reduce the evaluation complexity to $O(N)$ and even $O(1)$ if the previous qualities are memorized [20]. Despite changing the solution in small steps iteratively, these algorithms can, nevertheless, explore the solution space and interesting parts thereof quickly. The complete enumeration of such a “swap” neighborhood contains $N * (N-1)/2$ moves and, therefore, grows quickly with the problem size. This poses a challenge for solving larger instances of the QAP.

The QAP can also be used to model cases when there are more locations than facilities and also when there are more facilities than locations. In these cases dummy facilities with zero flows or dummy locations with a high distance can be defined.

QAPLIB

The quadratic assignment problem library [4] (QAPLIB) is a collection of benchmark instances from different contributors. According to their website[1], it originated at the Graz University of Technology and is now maintained by the University of Pennsylvania, School of Engineering and Applied Science. It includes the instance descriptions in a common format, as well as optimal and best-known solutions or lower bounds and consists of a total of 137 instances from 15 contributing sources which cover real-world as well as random instances. The sizes range from 10 to 256 although smaller instances are more frequent. All 103 instances between 12 and 50

[1] `http://www.seas.upenn.edu/qaplib/`

have been selected for this study with the exception of `esc16f`, which does not specify any flows.

10.1.2 Literature Review

Research on the QAP has a longer history, but is still very active. A few selected publications shall be briefly described here to give the reader an overview on the research efforts that have been conducted.

The first successful metaheuristic for the QAP was robust taboo search (RTS) and has been introduced in [20]. This algorithm addresses one of the main weaknesses of the standard tabu search (TS) algorithm. In tabu search it can often happen that the search trajectory returns to the same solution over and over. This cycling behavior is due to its rather deterministic nature and the attempt of Taillard was to randomize the tabu tenure and therefore make the trajectory more robust. Taillard also noted that some instances of the QAP, namely the `els19`, proved to be difficult to be solved. The search would get stuck in a certain sub-region of the search space and would be unable to escape as it would never climb high enough to leave it. He described a simple diversification strategy that would choose to make diversifying moves at certain times which were enough to solve the problem.

The genetic local search (GLS) algorithm has been introduced in [16]. It is a hybrid metaheuristic that combines elements of a genetic algorithm with local search. The authors note that local search is already able to solve the QAP quite well if it is applied multiple times, as will also be shown here. They developed a new crossover operator which they named *DistancePreservingCrossover*. It is a highly disruptive crossover that only preserves the common alleles and randomizes all other genes. The common alleles are then fixed and the local search is applied to find a new local optima for the other genes.

The performance of iterated local search was analyzed in [19]. In that work Stützle tested several strategies of an iterated local search algorithm. The local search is a first-improvement local search that would make use of so called "don't look bits" which limit the neighborhood of the local search to the interesting parts. He also uses concepts of variable neighborhood search in that he varies the degree of perturbation of the solution from which the local search is restarted after it has landed in a local optima. Finally, he hybridizes this algorithm with an evolution strategy that achieves very competitive results.

A survey and a little bit of history on solutions to some quadratic assignment problem instances is given in [9]. Drezner also proposed several new problem instances, which have however not found their way into the QAPLIB yet.

The topic of fitness landscape analysis on the quadratic assignment problem has initially been attempted in [17]. Merz and Freisleben described several measures such as autocorrelation and correlation length to measure landscape ruggedness and fitness distance correlation (FDC) to visualize the correlation between solution similarity (to the optimum) and quality. Problems are easier to solve when it holds that

the more similar a solution becomes to the optimum the better its quality. This can lead the search to the optimum instead of deceiving it.

The problem ruggedness has also been analyzed in [3]. Angel introduced a ruggedness coefficient and stated that the QAP appears to be a rather smooth problem which is beneficial for local search. Autocorrelation values recently have also been calculated exactly in [5]. Chicano et al obtain the autocorrelation function analytically instead of empirically.

In the following chapter, results and a comparison between different algorithms will be given on several instances of the quadratic assignment problem taken from the QAPLIB. The quality differences will be given on a scale between the best-known respectively optimal quality and an instance specific average. The reason is that the typically used relative difference between the obtained solution's fitness $f(s)$ and the best known solution f^*, given as $(f(s) - f^*)/f^*$, cannot be used to compare algorithm performance over multiple problem instances as a given instance might appear to be solved well, when actually only the range between an average solution's quality and the optimum is narrow. Therefore, we use the average fitness of the problem instance $\hat{f}$ to normalize this ratio and obtain a *scaled difference* as $(f(s) - f^*)/(\hat{f} - f^*)$. It is shown in [5] that $\hat{f}$ can be computed exactly for any instance of the QAP. The comparison of effort between the algorithms has been normalized to the computational task to evaluate a solution. The swap move that is often used can be calculated in O(N) and even O(1) [20]. If it is calculated in O(1) there are just 4 operations necessary, otherwise there are $4 * N$ operations required. The evaluation of a full solution requires N^2 operations. In this chapter *solution evaluation equivalents* denotes the amount of full solution evaluations taking into account that move evaluations need less operations. If not otherwise mentioned all results in this chapter are computed as an average of 20 repetitions of the same configuration.

10.2 Trajectory-Based Algorithms

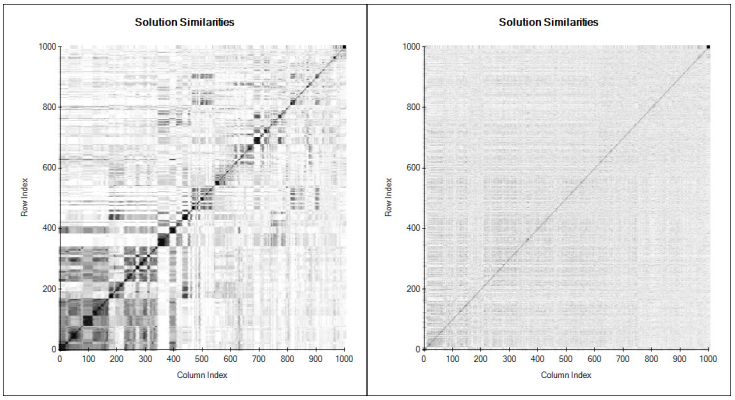
Trajectory-based algorithms attempt to move from one solution to the next through a number of smaller or larger steps. Different strategies exist to guide the movement. The most often used neighborhood in the QAP is the so called swap2 neighborhood where two indices in the permutation are randomly selected and exchanged. For a QAP solution this means that two facilities are placed in the location of the respective other facility. This move creates only a small change. All of the trajectory-based algorithms in this chapter make use of this type of move.

10.2.1 Local Search

Local search basically comes in two variants:

- **best-improvement:** Considers a number of moves at the same time. It attempts to make only the move with the maximum performance gain. It is sometimes also referred to a steepest or gradient-descent heuristic. If no more improving moves can be made the search knows to have reached a local optimum.
- **first-improvement:** Considers moves one after another and attempts to make each move that would improve the current solution. It does not follow the steepest gradient, but in some cases where the neighborhood is much larger than the number of steps to the local optimum first-improvement might be quicker to converge. It knows to have reached a local optimum when it has tried all moves and none constituted an improvement.

The performance of local search depends to a large degree on the number and quality of local optima that would attract the search. Typically a quadratic assignment problem contains many local optima that are scattered over the entire search space. Only some instances lead the search into similar local optima. In Figure 10.1 the similarity of local optima is shown for two selected instances. In the `els19` instance a randomly started local search often converges into similar local optima, while in the `bur26a` instance there are much more and scattered local optima.



(a) `els19` (b) `bur26a`

Fig. 10.1 Similarities of 1,000 randomly sampled local optima with each other for two selected problem instances. The darker an entry in this matrix the more similar two local optima are.

In Figure 10.2 it can be observed that the similarity of local optima reduces with the problem size which can be expected given the increase of the search space. However, the similarities change at a slightly slower rate (n^c) than search space ($n!$).

Maximum similarity reduces to about 20% for problems of size 50 which means that for a given local optimum there is at least one other optimum out of 999 that is 20% similar. Average similarity compares the similarity among all other local optima found.

As the results in Figure 10.3 show, small instances can be very deceptive for a local search. Although the problems belonging to the `rou` family are still small enough that the trajectory leads to the best solution the average trajectory ends up about 20% away from the optimum and 45% in the worst observed case. In general, the average quality of local optima varies greatly between the instances as well as the effort to reach them. It is also interesting to note that in the `lipa-b` instances the average fitness of local optima is rather high. The effort to find such an optimum using a best-improvement local search based on the swap2 neighborhood is steadily increasing with the problem size. While it takes on average 286 solution evaluation equivalents to find a local optimum in the `rou` family it takes about 4,837 solution evaluation equivalents for the `wil50` problem instance. Although such a trend exists, the difficulty of instances of the quadratic assignment problem cannot only be judged purely by problem size from the point of view of a local search algorithm.

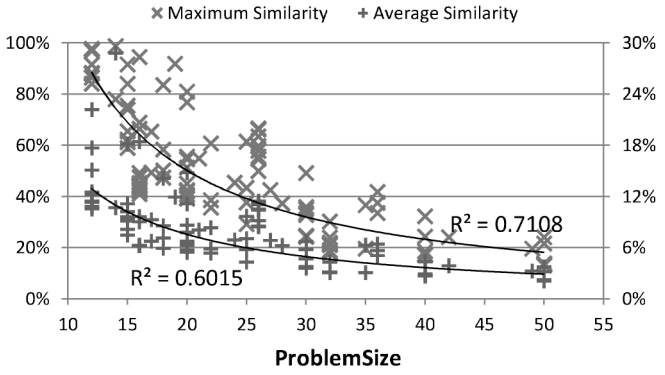


Fig. 10.2 Correlation of local optima similarity and problem size

The local search is often repeated to find a different and better local optimum in the next descent. Such a repetition might be performed from a completely random solution, or from a slight perturbation of the current local optimum in which case it is called an iterated local search (ILS). Table 10.1 shows the results from simple iterated first- and best-improving local search strategies. The algorithms were run with a maximum of 10,000 repeated starts. The search was stopped as soon as it reached the known optimum or best-known quality and the effort is given in the average number of evaluated solution equivalents. The search is restarted from a solution where on average about $\frac{1}{4}$ of the current solution is randomly perturbed. Tests were also conducted with a simple repeated local search with comparable, but slightly

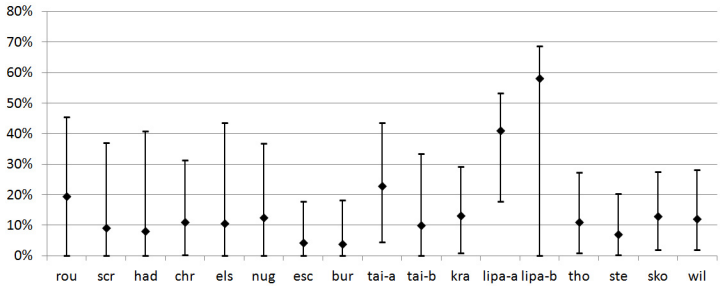


Fig. 10.3 Distribution of local optima with respect to their fitness values. The graphic shows the minimum, average, and maximum fitness for local optima found from 10,000 initial random samples.

worse results. The main difference between a basic iterated local search and a repeated local search is the required effort. Starting over from completely new random solutions requires considerable more effort to descend into a local optimum. When starting closer to the just found local optimum finding a new local and possibly also the global optimum is faster. In the case of iterated best-improvement local search the effort is about 180% higher when repeating from a new random solution, in first-improvement it is still about 52% higher. Nevertheless, there is the possibility that the trajectory gets stuck in a certain region. In our case however the strategy was very simple as the search always continued from the newly found local optimum and the scramble operation is able to modify even the whole permutation, although with only a very small probability.

It is also interesting to compare `lipa-a` and `lipa-b` instances with `tai-a` and `tai-b`. While the -b instances generally appear to be easier to solve `lipa-b` also requires a lot less effort, the effort for `tai-b` however remains roughly the same. Another interesting observation is that `els` of which the only instance is `els19` requires less effort in the best-improvement variant. This is noteworthy as it is the most significant out of three groups that showed this behavior. Overall, the roughly similar quality, but in general reduced effort of the first-improvement search makes it much more attractive to be used.

10.2.2 Simulated Annealing

In contrast to local search simulated annealing allows the trajectory to worsen with a certain probability. This counters the drawback of local search algorithms that get stuck in local optima as they are missing a strategy for escaping it. Simulated annealing uses a temperature parameter that governs the probability of accepting a worsening move. An often used cooling schedule is the exponential cooling where the probability to move from solution c to solution x in the case of a fitness

Table 10.1 Performance comparison of a simple first- and best-improving iterated local search on several instances of the QAPLIB

	First-improvement			Best-improvement		
Instance	Quality	Effort	Optimum	Quality	Effort	Optimum
bur	0.000%	105,409	100.0%	0.000%	232,074	100.0%
chr	0.475%	570,816	59.3%	0.322%	1,091,356	66.4%
els	0.000%	62,496	100.0%	0.000%	14,808	100.0%
esc	0.037%	84,323	95.9%	0.025%	241,453	96.8%
had	0.000%	5,324	100.0%	0.000%	22,223	100.0%
kra	0.066%	790,856	80.0%	0.127%	3,091,380	70.0%
lipa-a	10.083%	1,592,687	61.3%	8.695%	4,678,133	68.8%
lipa-b	0.000%	10,777	100.0%	0.000%	41,671	100.0%
nug	0.011%	205,228	96.0%	0.014%	535,476	95.7%
rou	0.023%	297,756	83.3%	0.074%	558,576	81.7%
scr	0.000%	28,307	100.0%	0.000%	145,315	100.0%
sko	0.803%	3,558,891	7.5%	0.845%	12,069,038	7.5%
ste	0.212%	2,126,174	33.3%	0.259%	7,214,283	28.3%
tai-a	3.843%	1,700,668	37.8%	4.195%	4,270,270	36.1%
tai-b	0.032%	1,506,763	65.6%	0.043%	4,827,457	60.6%
tho	0.407%	2,103,004	37.5%	0.522%	6,299,726	30.0%
wil	0.693%	4,473,760	0.0%	0.661%	15,415,083	0.0%
	0.849%	**713,103**	**76.4%**	**0.812%**	**2,063,833**	**76.6%**

deterioration is $p(x,c) < e^{\frac{f(c)-f(x)}{T}}$. As the temperature parameter T is reduced over time the search is less and less likely to accept large uphill moves.

As can be seen in the results of Table 10.2 the performance depends to a large degree on the start temperature. If the initial temperature is not chosen high enough the search will converge too quickly, if however the initial temperature is raised the search may converge into better regions of the search space. However, raising the temperature too far is not beneficial as the search would then only perform a random walk and waste CPU time. In the table the initial temperatures have been chosen according to the range between the average quality and the best-known quality. The search should accept with a probability of 1% a solution that worsens the current solution by 1%, 4%, 16%, or 32% of the fitness range. Note that in this case the best-known quality has been used, but if that is not available one could also use a suitable lower bound.

There are further parameters in simulated annealing that govern the speed of the cooling as well as the initial and final temperature. Cooling speed of 1 represents a linear cooling schedule while 0.01 and 0.00001 represent more rapid cooling schedules that would quickly prevent the acceptance of uphill moves. As shown in Figure 10.4 experiments regarding the variation of these parameters have shown that only the starting temperature has a significant influence on the final solution quality.

Detailed results are given in Table 10.3 for the configuration that worked best overall. It is interesting to see that simulated annealing is having difficulties in

Table 10.2 Performance of simulated annealing averaged over various configurations with respect to the chosen start temperature

T_0	Quality	Effort	Optimum
1%	7.155%	3,666,504	24%
4%	1.594%	2,507,105	53%
16%	**0.503%**	**1,760,454**	**71%**
32%	0.596%	2,158,207	69%

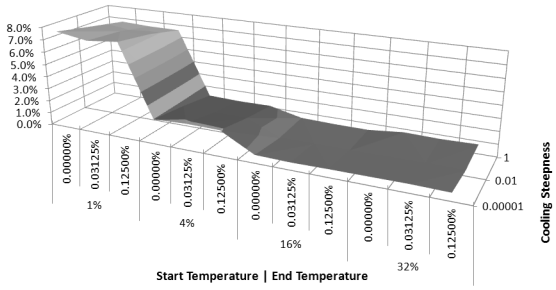


Fig. 10.4 Parameter range test for simulated annealing shows that start temperature has the largest effect on the performance. In this case the results are averaged over all repetitions of all tested instances.

solving the `els19` instance to optimality. On the other hand the algorithm has no trouble solving many of the other instances. The seemingly difficult `lipa-a` instances are solved with much more ease finding the optimum in all, but one case. Instance families that simulated annealing is not as well suited as a simple iterated local search are `bur`, `had`, `nug`, `scr`, and `tai-b`. In these instances the performance is worse despite sometimes a much larger effort that is required in obtaining good solutions. On the other hand simulated annealing is able to solve `chr`, `kra`, `rou`, `sko`, `ste`, `tai-a`, `tho`, and `wil` better than the simple iterated local search.

10.2.3 Tabu Search

Like simulated annealing tabu search also has a strategy for escaping local optima, which forces the search to make a move every time which is the best move that can currently be made. Naturally, this would likely return the search to the previous solution in the succeeding iteration so tabu search includes a memory that prevents to revert a move that has been made recently. This memory thus allows the search to explore greater parts of the search space.

The so called “standard" tabu search includes a simple memory that remembers the previously assigned location. Every move is declared *tabu* if it would reassign

Table 10.3 Performance of simulated annealing applied to several instances of the QAPLIB

	16%		
Instance	Average	Effort	Optimum
bur	0.247%	3,528,124	41.9%
chr	0.213%	2,137,507	67.1%
els	1.531%	4,901,637	15.0%
esc	0.000%	220,586	100.0%
had	0.183%	1,483,900	82.0%
kra	0.106%	1,544,613	85.0%
lipa-a	0.336%	839,576	98.8%
lipa-b	0.000%	802,769	100.0%
nug	0.035%	955,344	94.7%
rou	0.034%	1,095,646	81.7%
scr	0.000%	637,245	100.0%
sko	0.291%	1,962,898	25.0%
ste	0.082%	2,402,201	43.3%
tai-a	2.756%	2,101,099	38.3%
tai-b	0.345%	3,003,620	53.1%
tho	0.081%	2,072,757	50.0%
wil	0.406%	1,968,492	5.0%
	0.379%	**1,619,005**	**73.2%**

a location to a facility that it had already been assigned to within the last n iterations. Nevertheless sometimes a move should be made even though it is tabu, such as when it would find a new best solution. The so called aspiration condition ensures that such moves are considered nevertheless. In the “standard" tabu search the aspiration condition typically reconsiders moves that would find a new best solution. However, more advanced aspiration conditions exist that take the quality of a certain move into account and would undo a move if it improved the quality from the last time it was done. This can happen when other parts of the solution have changed significantly. Tabu search typically has one important parameter that governs the time that moves are kept tabu: the tabu tenure. The longer a certain move is kept tabu the less likely the search returns and the more it will diversify. However, if it is too long the search might be prevented to intensify and cannot explore the region around local optima. With less possible moves in the neighborhood to choose from the behavior approaches that of random search.

The results shown in Table 10.4 indicate the performance of standard tabu search applied to the already mentioned problem instances. The algorithm generally shows good search behavior, but it has problems to solve some instances such as from the `bur`, `els`, `had`, and `tai-b` family. As Taillard states in [20] in certain instances certain moves are never considered and thus the search is trapped in a sub-region of the search space. One possibility to counter this would be to increase the tabu tenure and thus force the search to diversify, however as already mentioned if the tenure becomes too large the search is only diversifying and will not descend and explore

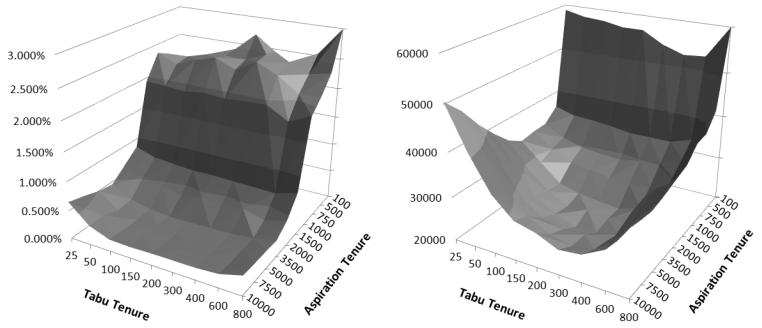
Table 10.4 Performance of a standard tabu search

	MaxIter = 2,000				MaxIter = 100,000	
	tenure = N		tenure = 2*N		tenure = 2*N	
Instance	Quality	Optimum	Quality	Optimum	Quality	Optimum
bur	2.772%	6.9%	2.341%	10.0%	2.121%	20.6%
chr	2.984%	16.8%	1.568%	23.2%	0.254%	69.6%
els	8.972%	0.0%	8.678%	10.0%	6.407%	10.0%
esc	0.320%	92.1%	0.214%	92.4%	0.015%	99.7%
had	1.539%	48.0%	1.265%	53.0%	1.069%	57.0%
kra	2.934%	11.7%	2.267%	21.7%	0.205%	90.0%
lipa-a	16.055%	40.0%	12.571%	55.0%	0.000%	100.0%
lipa-b	1.429%	97.5%	1.426%	97.5%	0.000%	100.0%
nug	0.821%	53.0%	0.644%	56.7%	0.057%	97.0%
rou	1.168%	36.7%	0.443%	68.3%	0.000%	100.0%
scr	1.570%	45.0%	0.100%	80.0%	0.000%	100.0%
sko	1.115%	2.5%	1.693%	0.0%	0.458%	12.5%
ste	1.326%	10.0%	1.152%	5.0%	0.169%	43.3%
tai-a	4.785%	10.6%	4.375%	19.4%	1.704%	46.7%
tai-b	8.258%	11.9%	7.559%	10.0%	7.325%	20.6%
wil	1.925%	0.0%	2.054%	0.0%	0.567%	0.0%
	2.991%	**39.2%**	**2.410%**	**44.1%**	**1.080%**	**69.6%**

interesting local optima. Therefore Taillard proposed the robust taboo search (RTS) that would randomize the tabu tenure to counter the probability the search stagnates. He also added a diversification strategy to explore still unseen parts of the search space. This algorithm has thus two main parameters the tabu tenure that governs the random variable from which the actual tenures are drawn and the aspiration tenure that defines a fixed number of iterations after diversifying moves are being made. These two parameters guide the intensifying and diversifying behavior of the search trajectory. Smaller tabu tenures lead to more intensification of close local optima, while larger tabu tenures will aim to diversify more. In a similar way a small aspiration tenure leads to more frequent diversification while larger aspiration tenures allow to intensify the search longer.

An extensive parameter study was conducted where for each of the 102 instances 90 configurations were tested. The configurations were chosen by combining each of 9 different tabu tenures (25, 50, 100, 150, 200, 300, 400, 600, and 800) with 10 different aspiration tenures (100, 500, 750, 1000, 1500, 2000, 3500, 5000, 7500, and 10000). The algorithm was allowed a total of 100,000 iterations. The averaged results over all instaces from the study can be seen in Figure 10.5 while the results of the average best configuration (tabu tenure of 200 and aspiration tenure of 7,500) are given in Table 10.5.

The robust taboo search is generally a very well working metaheuristic for problems of this size, however the parameter configuration is a bit harder. Both the tabu



(a) quality (b) iterations

Fig. 10.5 Parameter range test for robust taboo search shows that best results are achieved with a moderate tabu tenure and a higher aspiration tenure. Each point in the grid was evaluated 20 times by performing the algorithm on all 102 instances.

tenure and the aspiration tenure have a high and sometimes interdependent influence on the performance. Good parameter settings need to be chosen, otherwise the search will perform worse. Unfortunately it is rather difficult to give a good figure for tabu search's effort. The ability to evaluate certain moves in O(1) as described in [20] greatly increases the influence of other elements on the search effort. The effort is calculated by computing each move evaluation as 4 operations in Table 10.5.

10.2.4 Variable Neighborhood Search

Variable neighborhood search (VNS) is a further trajectory-based algorithm with another strategy to escape local optima. Instead of trying to simply "move on" as would tabu search do or escape by probability as would simulated annealing do the variable neighborhood search attempts to perform a set of changes to the current solution in the hope of perturbing it to a degree that allows different local optima to be reached. It is similar to an iterated local search, but it uses a more advanced strategy to continue the search. The strength of the modifications increases with the number of attempts: at first only small changes are made, but as the search remains unsuccessful in finding a better local optimum the strength of the change is increased. This creates a well working combination of fast local search behavior with a mutation-based diversification strategy. The performance of VNS is also shown in Table 10.5 in which the algorithm was run for 500 full cycles of shaking operators. The results shown are the averages of those runs. It finds comparable solutions to a first-improvement local search although it has been given much less time.

Table 10.5 Performance of the robust taboo search (RTS) and the variable neighborhood search (VNS)

	RTS				VNS		
Instance	Quality	Iterations	Effort	Optimum	Quality	Effort	Optimum
bur	0.112%	37,964	73,008	88.8%	0.000%	29,753	100.0%
chr	0.173%	36,709	69,834	75.4%	0.379%	210,424	59.3%
els	0.000%	23,805	45,103	100.0%	0.000%	14,187	100.0%
esc	0.025%	6,092	11,802	97.1%	0.021%	25,067	97.9%
had	0.228%	17,730	33,435	90.0%	0.000%	4,213	100.0%
kra	0.176%	37,061	71,690	91.7%	0.708%	548,332	53.3%
lipa-a	0.000%	11,712	22,833	100.0%	13.795%	630,936	50.0%
lipa-b	0.000%	1,106	2,156	100.0%	0.000%	32,010	100.0%
nug	0.000%	5,362	10,274	100.0%	0.103%	123,707	84.7%
rou	0.001%	9,244	17,543	98.3%	0.247%	144,587	70.0%
scr	0.000%	1,752	3,314	100.0%	0.000%	21,900	100.0%
sko	0.109%	58,454	114,423	55.0%	0.667%	1,259,354	20.0%
ste	0.015%	41,464	80,625	91.7%	0.097%	461,140	70.0%
tai-a	1.042%	50,626	98,112	62.8%	4.025%	597,374	27.8%
tai-b	0.116%	41,990	81,548	73.8%	0.048%	251,736	86.9%
tho	0.030%	51,952	101,211	57.5%	0.596%	843,418	17.5%
wil	0.137%	86,947	170,417	40.0%	0.702%	1,684,106	0.0%
	0.159%	**24,836**	**47,895**	**86.6%**	**1.033%**	**244,711**	**75.1%**

It applies swap2, swap3, scramble, inversion, insertion, translocation, and translocation-inversion as shaking operators in this order [2].

10.3 Population-Based Algorithms

Population-based algorithms attempt to make use of a larger number of solutions that are evolved over time. Often in optimization it is beneficial to explore the search space more rigorously in order to find better solutions. There are a number of strategies which involve various degrees of replacement and elitism, that control how the algorithms would discard a solution and instead accept a new one. For instance, in a genetic algorithm with generational replacement the newly evolved population always fully replaces the old population, even if it was of worse quality. Often only one single best individual is retained from the old population. This strategy requires the crossover operator to combine relevant genetic information of high quality individuals. In the offspring selection genetic algorithm however, the replacement strategy puts more pressure towards fulfilling this requirement: The search accepts new children only if they outperform their respective parents. This strategy often works better than a standard genetic algorithm, however in each generation many more children are produced and the convergence is slowed. At the same time this allows

a more thorough exploration of the search space which should yield higher quality results.

In contrast to trajectory-based algorithms, the introduction of a population should benefit in those cases when the fitness landscape has many different basins of attraction [18]. This does correlate with the number of local optima, but not fully as local optima might also be quite close to another and form valleys of attraction (it is believed that e.g. the traveling salesman problem does generate such a "big-valley" landscape in which good solutions are also quite similar to each other [6]). Population-based search, at least in the beginning, is able to discover multiple such basins and explore them simultaneously. Trajectory-based algorithms always only explore to the root of one basin at a time and then have to find their way to the next basin.

As was observed, the behavior of the standard tabu search on the `els19` resulted in a search trajectory that got stuck in a confined part of the search space. The same problem instance is however comparably easy to solve for a genetic algorithm. However, as mentioned, population-based algorithms are only successful if the distributed information on relevant genes can be combined in one chromosome through the process of survival of the fittest and crossover. Especially when solving the QAP it can be observed that this process is not always successful.

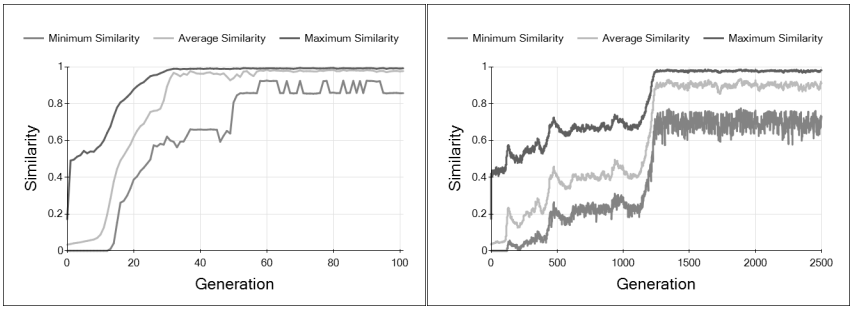
10.3.1 Genetic Algorithm

The performance of the genetic algorithm depends to a large part on whether the crossover operator can combine the relevant genes from multiple individuals in a new offspring. However, as can be seen in a parameter study involving various crossover operators which are generally deemed suitable, various mutation operators, and mutation probabilities the genetic algorithm population cannot effectively converge in optimal regions of the search space. The search converges prematurely and stagnates at a certain level to the known optimum with much of the diversity lost. At this point crossover is not relevant anymore and only mutation may introduce new alleles into the population. Table 10.6 shows the average performance of standard genetic algorithm with a population size of 500, and full generational replacement. For the tests with roulette-wheel selection the algorithm was run only for 5,000 generations, but achieved on average better results than tournament selection run for 10,000 generations. This shows that the lower selection pressure is able to maintain genetic diversity longer which increases the chance of combining it in a single solution.

As Figure 10.6 shows the genetic diversity in the population can be lost as early as generation 30 if selection pressure is very high. After the search has converged the continued evolution is mainly driven by random mutations with a low probability and therefore highly inefficient. Using a selection operator that excerts less selection pressure such as roulette-wheel in this case shows that convergence can be

Table 10.6 Performance of a standard genetic algorithm with partially matched crossover (PMX) [11], swap2 manipulation and 15% mutation probability

	5-Tournament			Roulette		
Instance	Quality	Effort	Optimum	Quality	Effort	Optimum
bur	1.221%	4,897,518	1.9%	1.338%	2,494,196	0.6%
chr	5.569%	4,765,084	4.6%	4.245%	2,212,262	15.4%
els	0.460%	1,508,229	70.0%	0.743%	2,400,715	5.0%
esc	1.307%	1,006,638	80.3%	1.089%	543,318	80.0%
had	1.505%	2,645,654	48.0%	0.984%	1,126,099	60.0%
kra	9.021%	4,990,500	0.0%	7.745%	2,493,571	1.7%
lipa-a	38.429%	4,928,337	1.3%	3.695%	2,388,408	5.0%
lipa-b	50.978%	4,435,188	11.3%	6.410%	2,158,538	17.5%
nug	6.122%	4,791,288	4.0%	4.488%	2,314,742	9.0%
rou	10.605%	4,990,500	0.0%	6.970%	2,137,933	15.0%
scr	4.701%	4,243,156	15.0%	3.702%	2,197,747	21.7%
sko	10.211%	4,990,500	0.0%	7.410%	2,495,500	0.0%
ste	5.342%	4,990,500	0.0%	7.326%	2,495,500	0.0%
tai-a	17.875%	4,962,855	0.6%	4.003%	2,408,973	3.9%
tai-b	3.334%	4,305,529	13.8%	2.773%	2,203,563	12.5%
tho	8.795%	4,990,500	0.0%	6.684%	2,495,500	0.0%
wil	9.227%	4,990,500	0.0%	0.459%	2,495,500	0.0%
	8.737%	**3,978,419**	**20.4%**	**7.315%**	**1,969,243**	**23.1%**



(a) 5-Tournament (b) Roulette

Fig. 10.6 The chart displays the average similarities in the population. It can be seen that the genetic diversity in the population of the GA is lost by generation 30 already when optimizing the `lipa30a` instance with high selection pressure. If the selection pressure is less the population takes longer to converge allowing the search to explore more of the search space.

prolonged for many generations. Selection pressure generally refers to the ratio of selecting better solutions to selecting worse solutions.

10.3.2 Offspring Selection Genetic Algorithm

The offspring selection genetic algorithm (OSGA) [1, 2] is an advanced and more robust variant of a genetic algorithm. Instead of relying on the undirected stochastic nature of the crossover and mutation operators it introduces another selection criteria for the newly created offspring. If the performance of the offspring surpasses that of its parents it is admissible to the next population, if however it is worse than the parents (the better parent) the offspring is discarded. This places a variable evolutionary pressure on the population forcing it to produce better and better offspring. The visible effect is that the OSGA performs much better than the standard GA in both convergence speed and quality. Still, the performance is not entirely satisfying and it is likely possible that even in this case the crossover operator is not able to combine the relevant genetic information. The performance of OSGA is given in Table 10.7. The SuccessRatio was set to 0.5 and the mutation probability was set to 25%. The algorithm selects one parent with tournament selection and a group size of 3 and the other parent randomly. In this case a variant was used where the individuals that remained after the SuccessRatio was filled were not selected from the pool of unsuccessful individuals, but were randomly selected from the previous population.

Table 10.7 Performance of the offspring selection genetic algorithm (OSGA)

Instance	Quality	Effort	Optimum
bur	0.479%	388,609	11.3%
chr	1.932%	416,956	27.1%
els	0.000%	63,000	100.0%
esc	1.033%	117,199	84.1%
had	0.590%	167,030	62.0%
kra	4.742%	462,953	0.0%
lipa-a	31.173%	626,594	12.5%
lipa-b	32.062%	557,218	47.5%
nug	1.987%	378,197	23.3%
rou	3.114%	406,432	20.0%
scr	0.742%	217,933	63.3%
sko	6.876%	669,170	0.0%
ste	2.332%	500,843	0.0%
tai-a	11.810%	528,069	14.4%
tai-b	1.319%	376,965	27.5%
tho	4.523%	477,328	0.0%
wil	4.787%	608,935	0.0%
	5.013%	**369,409**	**34.3%**

As expected from previous applications of OSGA [2] the algorithm could greatly improve the results compared to the standard genetic algorithm. However, a few exceptions are interesting to note regarding instances of type `lipa-a` and `lipa-b`,

as well as tai-a and wil. In these instances OSGA was able to find the optimal or best known solution more often, however the average quality is worse than in the standard GA. It can also be seen that these are instances where the standard genetic algorithm with tournament selection also struggled much more. A selection scheme with less selection pressure seems to be the more favorable approach, meaning that the search needs to diversify with a much greater degree. A similar observation can be made when the results from standard tabu search and robust taboo search are compared.

10.4 Hybrid Algorithms

The goal of hybridizing algorithms is to combine good properties of a number of algorithms in one strategy. From the previous experience on performance indicators it can be derived that trajectory-based algorithms with their step-wise local search behavior appear to work very well given a certain diversification strategy.

The intent of this chapter is to show and analyze possible performance improvements by combining population-based and trajectory-based algorithms to so called memetic algorithms. The hope is that such algorithms inherent the good search properties of the strategies that they combine and are thus more universally applicable than strategies that focus solely on a search trajectory or on the combination through crossover.

10.4.1 Memetic Algorithms

Memetic algorithms combine the advantages of genetic algorithms with local search behavior. They are called “memetic" because the solutions are changed during the generation as opposed to only in the time between generations. The GLS algorithm mentioned in Section 10.1.2 is such a memetic algorithm. However, naturally one needs to take care that in the design of such an algorithm the population does not converge too quickly, as it draws its diversifying power from the genetic diversity in the population. The problem known as *genetic drift* could be even worse in this case when the local search decends into highly similar local optima. Here we compare the performance of GLS with that of a simple genetic algorithm that is adapted with local search.

In the results were the genetic algorithm was combined with a local search we can see that the search could indeed be improved over just the genetic algorithm or just the local search. The best-known solution was found only slightly more often than in the results of ILS (77.2% vs 76.6%), but the average quality could be significantly improved with an effort in between the first-improvement and best-improvement ILS. The algorithm did however not improve in all instances, but the improvement can mainly be attributed to instances where the ILS gave worse results such as lip-a and tai-a. It is interesting to see that GA+LS for example could

Table 10.8 Performance of GLS and GA+LS compared

	GLS			GA+LS		
Instance	Quality	Effort	Optimum	Quality	Effort	Optimum
bur	0.000%	114,751	100.0%	0.050%	343,540	87.5%
chr	0.009%	627,206	98.6%	0.269%	352,540	62.9%
els	0.000%	57,803	100.0%	0.649%	96,586	55.0%
esc	0.004%	91,802	99.4%	0.002%	80,480	99.7%
had	0.000%	19,718	100.0%	0.000%	16,114	100.0%
kra	0.000%	797,324	100.0%	0.132%	428,414	85.0%
lipa-a	0.000%	1,196,204	100.0%	0.300%	334,641	98.8%
lipa-b	0.000%	64,669	100.0%	0.000%	48,404	100.0%
nug	0.000%	166,994	100.0%	0.081%	130,326	87.7%
rou	0.000%	549,678	100.0%	0.259%	103,860	65.0%
scr	0.000%	56,966	100.0%	0.010%	59,681	98.3%
sko	0.081%	9,733,115	65.0%	0.367%	1,328,217	35.0%
ste	0.012%	2,905,395	91.7%	0.110%	743,858	55.0%
tai-a	1.720%	6,431,136	52.2%	3.077%	914,393	22.8%
tai-b	0.000%	403,488	100.0%	0.054%	403,152	86.3%
tho	0.042%	8,894,650	55.0%	0.268%	742,103	22.5%
wil	0.146%	17,108,610	40.0%	0.391%	1,778,433	10.0%
	0.158%	**1,444,752**	**93.1%**	**0.378%**	**334,290**	**77.2%**

not succeed in the `els19` instance which the ILS perfectly solved. The hybrid inherited the problems of the genetic algorithm in this instance. GLS on the other hand shows good performance throughout the instances often solving them to optimality. Among all tested algorithms it did find the optimal or best-known solution most often.

10.5 Summary

Two algorithms achieved very good results overall: Robust taboo search (RTS) and genetic local search (GLS). However, the summarized results of the best performing metaheuristic for each problem instance family as shown in Table 10.9 suggest that for certain problem instances there are different algorithms that emerge as best. Algorithms in the table were deemed to perform better than another when they find on average better quality solutions as the maximum allowable effort in each algorithm was comparable. If two or more algorithms find equally good solutions, then the better performing algorithm is the one that is requiring less actual effort. Variable neighborhood search is not to be underestimated in situations where also the repeated or iterated local search is delivering good results. Also an implementation of tabu search that is oriented on the template described in [12] outperforms the robust variant on several instances. Usually both perform equally well, but the

standard implementation uses less iterations. This shows the trade-off that robust taboo search makes in that it is on average performing much better. Finally, genetic local search showed the best overall results, but also emerged as the best algorithm only in some cases.

Table 10.9 Summary table that lists the best performing results

Instance	Best performing	Instance	Best performing
bur	VNS	rou	Standard TS
chr	Genetic LS	scr	Standard TS
els	VNS	sko	Genetic LS
esc	Genetic LS	ste	Genetic LS
had	VNS	tai-a	Robust TS
kra	Genetic LS	tai-b	Genetic LS
lipa-a	Standard TS	tho	Robust TS
lipa-b	Standard TS	wil	Robust TS
nug	Robust TS		

10.6 Conclusions

As we have shown in this work the quadratic assignment problem is an interesting problem despite its age. Some standard metaheuristics from the first days do not perform as well and several modifications need to be made. Metaheuristics of later generations such as variable neighborhood search do perform very well out of the box, but the clever combination of population-based and trajectory-based approaches can lead to very good results overall. It is interesting that the diversity in the characteristics of these instances is reflected in the heterogeneity of the applied algorithms and while there exist overall well-performing algorithms they are often not the best algorithm for every instance. The results also suggest that, despite the overall strength of RTS and GLS there is no single best algorithm for all the instances. To design better algorithms is one possible approach to continue making progress in solving the QAP, but more importantly, research needs to be done to decide which algorithm to choose for a concrete and previously unobserved instance. As more experiments are performed in such studies the knowledge on the performance on individual algorithms is increasing. The topic of fitness landscape analysis seems to be most promising to extract problem instance characteristics that could be used to indicate which algorithm to choose and constitutes an interesting base for further research.

Acknowledgements. The work described in this chapter was done within the Josef Ressel-Centre HEUREKA! for Heuristic Optimization sponsored by the Austrian Research Promotion Agency (FFG).

References

1. Affenzeller, M., Wagner, S.: Offspring selection: A new self-adaptive selection scheme for genetic algorithms. In: Ribeiro, B., Albrecht, R.F., Dobnikar, A., Pearson, D.W., Steele, N.C. (eds.) Adaptive and Natural Computing Algorithms. Springer Computer Series, pp. 218–221. Springer (2005)
2. Affenzeller, M., Winkler, S., Wagner, S., Beham, A.: Genetic Algorithms and Genetic Programming - Modern Concepts and Practical Applications. Numerical Insights. CRC Press (2009)
3. Angel, E., Zissimopoulos, V.: On the landscape ruggedness of the quadratic assignment problem. Theoretical Computer Science 263(1-2), 159–172 (2001)
4. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB - A quadratic assignment problem library. Journal of Global Optimization 10(4), 391–403 (1997)
5. Chicano, F., Luque, G., Alba, E.: Autocorrelation measures for the quadratic assignment problem. Applied Mathematics Letters 25, 698–705 (2012)
6. Czech, Z.J.: Statistical measures of a fitness landscape for the vehicle routing problem. In: Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, IPDPS 2008 (2008)
7. de Carvalho Jr., S.A., Rahmann, S.: Microarray layout as quadratic assignment problem. In: Proceedings of the German Conference on Bioinformatics (GCB). Lecture Notes in Informatics, vol. P-83 (2006)
8. Drezner, Z.: Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. Computers & Operations Research 35, 717–736 (2008)
9. Drezner, Z., Hahn, P.M., Taillard, E.D.: Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult to solve for meta-heuristic methods. Annals of Operations Research 139, 65–94 (2005)
10. Elshafei, A.N.: Hospital layout as a quadratic assignment problem. Operational Research Quarterly 28(1), 167–179 (1977)
11. Fogel, D.: An evolutionary approach to the traveling salesman problem. Biological Cybernetics 60, 139–144 (1988)
12. Glover, F.: Tabu search – part I. ORSA Journal on Computing 1(3), 190–206 (1989)
13. Hahn, P.M., Krarup, J.: A hospital facility layout problem finally solved. Journal of Intelligent Manufacturing 12, 487–496 (2001)
14. James, T., Rego, C., Glover, F.: Multistart tabu search and diversification strategies for the quadratic assignment problem. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 39(3), 579–596 (2009)
15. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica, Journal of the Econometric Society 25(1), 53–76 (1957)
16. Merz, P., Freisleben, B.: A genetic local search approach to the quadratic assignment problem. In: Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 465–472. Citeseer (1997)
17. Merz, P., Freisleben, B.: Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Transactions on Evolutionary Computation 4(4), 337–352 (2000)
18. Pitzer, E., Affenzeller, M., Beham, A.: A closer look down the basins of attraction. In: UK Conference on Computational Intelligence (2010) (in press)
19. Stutzle, T.: Iterated local search for the quadratic assignment problem. European Journal of Operational Research 174, 1519–1539 (2006)
20. Taillard, E.D.: Robust taboo search for the quadratic assignment problem. Parallel Computing 17, 443–455 (1991)