Submission to Annals of OR (revised June 2002)

Applying an extended Guided Local Search to the Quadratic Assignment Problem

Patrick Mills, Edward Tsang and John Ford
Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK
Phone: +44 01206 87{2771,2774,2787}
E-mail: {millph,edward,fordj}@essex.ac.uk
World Wide Web: http://cswww.essex.ac.uk/CSP/

Abstract. In this paper, we show how an extended Guided Local Search (GLS) can be applied to the Quadratic Assignment Problem (QAP). GLS is a general, penalty-based meta-heuristic, which sits on top of local search algorithms, to help guide them out of local minima. We present empirical results of applying several extended versions of GLS to the QAP, and show that these extensions can improve the range of parameter settings within which Guided Local Search performs well. Finally, we compare the results of running our extended GLS with some state of the art algorithms for the QAP.

Keywords: local search, meta-heuristics, quadratic assignment problem

Guided Local Search (GLS) [29] has been applied to a number of problems, including the SAT problem [18], the weighted MAX-SAT problem [18], the vehicle routing problem [10], BT's workforce scheduling problem [28], the radio link frequency assignment problem [30], function optimisation [31] and the travelling salesman problem [32].

GLS is a general meta-heuristic that sits on top of local search procedures and helps them escape from local minima. GLS can be seen as a generalisation of the GENET neural network [27,5,6] for solving constraint satisfaction problems and optimisation problems. Recently, it has been shown that GLS can be put on top of a specialised Genetic Algorithm, resulting in the Guided Genetic Algorithm (GGA) [16]. GGA has been applied to a number of problems, including the processor configuration problem [11,12,14], the generalised assignment problem [13] and the radio link frequency assignment problem [15,17]. In this paper, we show how GLS and some extensions of GLS can be successfully applied to the Quadratic Assignment Problem.

1 The Quadratic Assignment Problem

The Quadratic Assignment Problem [4] is one of the hardest groups of problems in combinatorial optimisation, with many real world applications and has been the focus of a lot of successful research into heuristic search methods. The problem can be formally stated as in equation (1).

$$\min_{\pi} g(\pi), where\ g(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi_i \pi_j} \tag{1}$$

where:

- n is the size of the problem (i.e. number of facilities or locations),
- π is a permutation, where π_i is the i^{th} element in permutation π,
- a and b are the n x n distance and flow matrices.

The problem is to find a permutation π (which represents which facilities are placed at which locations), which minimises the sum of the distance times the flow between different facilities. Each element a_{ij} of the matrix a, represents the distance between location i and location j. The element $b_{\pi_i \pi_j}$ represents the flow between facilities π_i and π_j. When a_{ij} is multiplied by $b_{\pi_i \pi_j}$, the cost of placing facility π_i at location i and facility π_j at location j, is obtained. Thus, by summing all the terms together, the total cost of the whole permutation of location-facility assignments is obtained.

Both exact and heuristic algorithms have been proposed for solving the Quadratic Assignment Problem (for a survey, see [19]). The exact algorithms have the disadvantage that they can only solve relatively small QAPs ($n \leq 20$), whereas the heuristic methods can deal with much larger problems. The heuristic methods, which have been used to solve the QAP, include Robust Tabu Search [23,24], Reactive Tabu Search [2], Simulated Annealing [33,26], a Genetic Hybrid Algorithm [9], ant algorithms [22, 25] and various others [1]. In this paper, we show how Guided Local Search (also a heuristic method) can be applied to the Quadratic Assignment problem, and present empirical results showing two extensions of Guided Local Search which can increase the range of parameters under which good results are obtained.

2 Guided Local Search

Guided local search (GLS) (see [29] for a more detailed description) is a metaheuristic, which sits on top of a local search algorithm. When the given local search algorithm settles in local optimum, GLS changes the objective function, by increasing penalties present in an augmented objective function, associated to *features* contained in that local optimum. The local search then continues to search using the augmented objective function, which is designed to bring it out of the local optimum.

Solution *features* are defined to distinguish between solutions with different characteristics, so that bad characteristics can be penalised by GLS, and hopefully removed by the local search algorithm. The choice of solution features therefore depends on the type of problem, and also to a certain extent on the local search algorithm. Each feature, f_i defined must have the following components:

- An Indicator function, I_i indicating whether the feature is present in the current solution or not:

$$I_i(s) = \begin{cases} 1, & \text{solution } s \text{ has property i} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

- A cost function $c_i(s)$, which gives the cost of having the feature present in a solution.
- A penalty p_i, initially set to 0, used to penalise occurrences of the feature, in local minima.

2.1 Selective Penalty Modifications

When the Local Search algorithm returns a local minimum, s, GLS penalises (increments the penalty of the feature) all the features present in that solution which have maximum utility, $util(s,f_i)$, as defined in equation (3).

$$util(s, f_i) = I_i(s) \cdot \frac{c_i(s)}{1 + p_i} \tag{3}$$

The idea is to penalise features, which have high costs first, although the utility of doing so decreases as the feature is penalised more and more times.

2.2 Augmented Cost Function

GLS uses an augmented cost function (4), to allow it to guide the Local Search algorithm out of the local minimum, by penalising features present in that local minimum. The idea is to make the local minimum more costly than the surrounding search space, where these features are not present.

$$h(s) = g(s) + \lambda \cdot \sum_{i=1}^{n} I_i(s) \cdot p_i \tag{4}$$

The parameter λ may be used to alter the intensification of the search for solutions. A higher value for λ will result in a more diverse search, where plateaus and basins in the search are searched less carefully; a low value will result in a more intensive search for the solution, where the basins and plateaus in the search landscape are searched with more care. Generally, a value of lambda, which is near to the average change in objective function after a move, will work well.

2.3 Local Search for the QAP

The Quadratic Assignment Problem (QAP) can be formulated as a local search algorithm, using the objective function defined in equation (1), and searching the space of possible permutations. The local search neighbourhood is simply the set of possible permutations resulting from the current permutation with any two of the elements transposed.

2.4 Efficient Local Search and Neighbourhood Updating for the QAP

The new value of the objective function after a swap can be efficiently incrementally updated in approximately O(n^2) time using (5) and (6) (for asymmetric QAPs, see [24]) or (7) and (8) (for symmetric QAPs, the symmetry in the matrices can be taken advantage of to speed up neighbourhood updating by a factor of about 4, see [2]).

$$\Delta g(\pi, r, s) = \begin{array}{l} (a_{rr} - a_{ss})(b_{\pi_s \pi_s} - b_{\pi_r \pi_r}) + (a_{rs} - a_{sr})(b_{\pi_s \pi_r} - b_{\pi_r \pi_s}) + \\ \displaystyle\sum_{k=1, k \neq r,s}^{n} ((a_{kr} - a_{ks})(b_{\pi_k \pi_s} - b_{\pi_k \pi_r}) + (a_{rk} - a_{sk})(b_{\pi_s \pi_k} - b_{\pi_r \pi_k})) \end{array} \tag{5}$$

$$\forall u,v \bullet u,v \neq r,s : \Delta g(\pi', u, v) = \begin{array}{l} \Delta g(\pi, u, v) + (a_{ru} - a_{rv} + a_{sv} - a_{su})(b_{\pi'_s \pi'_u} - b_{\pi'_s \pi'_v} + b_{\pi'_r \pi'_v} - b_{\pi'_r \pi'_u}) + \\ (a_{ur} - a_{vr} + a_{vs} - a_{us})(b_{\pi'_u \pi'_s} - b_{\pi'_v \pi'_s} + b_{\pi'_v \pi'_r} - b_{\pi'_u \pi'_r}) \end{array} \tag{6}$$

$$\Delta g(\pi, r, s) = 2 \sum_{k=1, k\neq r,s}^{n} (a_{rk} - a_{sk})(b_{\pi_s \pi_k} - b_{\pi_r \pi_k}) \tag{7}$$

$$\forall u, v \bullet u, v \neq r, s : \Delta g(\pi', u, v) = \begin{array}{c} \Delta g(\pi, u, v) + \\ 2(a_{ru} - a_{rv} + a_{sv} - a_{su})(b_{\pi'_s \pi'_u} - b_{\pi'_s \pi'_v} + b_{\pi'_r \pi'_v} - b_{\pi'_r \pi'_u} \end{array} \tag{8}$$

$$\Delta g(\pi, r, s) = 2 \sum_{k=1, k\neq r,s}^{n} (b_{\pi_s \pi_k} - b_{\pi_r \pi_k}) \tag{9}$$

$$\forall u, v \bullet u, v \neq r, s : \Delta g(\pi', u, v) = \Delta g(\pi, u, v) + 2(b_{\pi'_s \pi'_u} - b_{\pi'_s \pi'_v} + b_{\pi'_r \pi'_v} - b_{\pi'_r \pi'_u}) \tag{10}$$

$$\Delta h(\pi, r, s) = \Delta g(\pi, r, s) + \lambda \cdot \left(\left(p_{r,\pi s} + p_{s,\pi r} \right) - \left(p_{r,\pi r} + p_{s,\pi s} \right) \right) \tag{11}$$

Where:
a & *b* are the distance and flow matrices, *n* is the problem size (i.e. the number of facilities or locations)
π' = the permutation π with elements *r* and *s* swapped
$\Delta h(\pi,i,j)$ = change in augmented cost h of permutation π, after the elements *i* and *j* have been swapped
$\Delta g(\pi,i,j)$ = change in cost *g* of permutation π, after the elements *i* and *j* have been swapped
$p_{i,\pi i}$=the penalty when the i^{th} element of permutation π is assigned the value π_i.

In addition to this, for Taillard's Grey density problems the neighbourhood may be restricted to swapping elements from the first m values in the permutation with the last n-m values in the permutation and may be calculated and updated more efficiently using equations (9) and (10) (as explained in [24]). The augmented cost may also be efficiently updated, using these equations, together with equation (11).

2.5 Features for the QAP

There is only one obvious choice for the feature set: facility-location assignments[1]. For each facility-location assignment π_i=*v*, there is an associated penalty $p_{i,v}$. Obviously the feature π_i=*v* is only present in a solution π if the *i*th element of π is *v*. All the penalties can be kept in a matrix of size *n* by *n* and the augmented objective function can be updated efficiently. The cost of a particular facility-location assignment is the sum of the constituent parts of the objective function that it is involved in (12).

$$Cost(i, \pi i) = \sum_{j=1}^{n} a_{i,j} b_{\pi i, \pi j} \tag{12}$$

2.6 A basic GLS for the QAP

In Figure 1, we show pseudo code for a basic GLS for the QAP, which we call GLSQAP. In line 1, λ is set to an initial value (we found the formula shown gave good results, by experimentation, when λ_{coeff} is 1). In line 2, we set π to a random initial start point, by randomly shuffling[2] the permutation (this gives equal probability of using any given permutation as the start point). Lines 3-10 iteratively apply local search to the current solution π, using the augmented objective function h. Line 6 calls the local search with the current solution, the original objective function and the augmented objective function as parameters. Line 7 selects features with maximum utility in the current local minimum returned by the local search to penalise. Line 8 increases the amount of penalty associated with each of those features in the augmented objective

---

[1] We did try to use pairs of facility-location assignments, with the flow between the facilities as the cost, but this meant there were a too many (N^4) features to store and incremental updating of the neighbourhood also became too expensive (more than 4 times slower) for larger problems, even with very "lazy" schemes for neighbourhood updating.

[2] This was implemented by the random_shuffle() function from the C++ standard template library (see [2], p538 for details of the random_suffle() function).

function. This continues until the termination criteria (line 10) is met (in our experiments in this paper, when 1000 * n swaps of elements of the permutation have occurred). Finally (line 11), GLS returns π^*, the permutation with the best cost found, during the search.

In Figure 2, we show pseudo code for a basic local search algorithm to be used with GLSQAP. This takes as parameters, a starting permutation π, an augmented objective function *h*, and the original objective function *g*. Line 1, sets the sideways count to zero. This is used to count the number of sideways moves (moves to solutions of equal augmented cost *h*). Line 2-10 iteratively modify the solution until a local minimum is found. This is defined to be (line 2) when there are no downwards and no sideways moves available (or if the maximum number of consecutive sideways moves, 2 in this paper[3], has been exceeded). Line 3 modifies the current permutation π, by swapping the elements, which result in the permutation with the lowest augmented cost. Lines 4-7 keep track of how many consecutive sideways moves have been made. Line 8 records the current solution π as the current best solution π^*, if it is the lowest cost solution found so far, w.r.t. the original cost function *g*. Line 10 returns the local minimum solution π.

GLSQAP(λ_{coeff})
{
1. $\lambda = \dfrac{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \times \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}}{n^4} \cdot \lambda_{coeff}$
2. $\pi^* = \pi$ = *randomly generated permutation of the values [1..n]*
3. ***do***
4. {
5. *//π is a permutation, the second parameter is the augmented objective function and //the third parameter is the original objective function*
6. $\pi = LocalSearch(\pi, \pi^*, g + \lambda \cdot \sum_{i=1}^{n} p_{i,\pi i}, g)$
7. ***foreach*** *(i in {1..n}), such that* $Cost(i,\pi_i) / (1 + p_{i,\pi i})$ *is maximised*
8. $p_{i,\pi i} = p_{i,\pi i} + 1$
9. }
10. ***while*** *(Not termination criteria)*
11. ***return*** π^*
}

Figure 1: Pseudo code for GLSQAP

LocalSearchQAP(π, π^*, *h*, *g*)
{
1. *sideways_count = 0*
2. ***while*** *((there is a downwards move w.r.t.* $h(\pi)$*) or (there is a sideways move w.r.t.* $h(\pi)$ *and sideways_count < 2) and termination criteria is not met)*
{
3. $\pi = \pi$ *with the elements* π_i *and* π_j *swapped such that* $\Delta h(\pi$ *with* π_i *and* π_j *swapped) is minimised (ties are broken randomly)*
4. ***if*** $(\Delta h(\pi) == 0)$
5. *sideways_count = sideways_count + 1*
6. ***else***
7. *sideways_count = 0*
8. *If* $(g(\pi) < g(\pi^*))\ \pi^* = \pi$
9. }

[3] This value was found to work well previously in [6], although this may not be the optimal value for this problem.

10. ***return*** π
}

Figure 2: Pseudo code for a basic local search for the QAP

3 Guided Local Search extensions

Whilst applying Guided Local Search to the QAP, we tried various schemes in an attempt to further improve Guided Local Search and try to understand why those schemes might work.

3.1 Adding aspiration moves to GLS

Aspiration criteria (as used in the tabu search framework, [7,8]) are conditions under which a move is allowed, even when it would normally be tabu, usually when it will give rise to a new best solution. Intuitively, this is a good idea, since it would be stupid to avoid making a move just because it was tabu, if it gave us a new best solution. In GLS, we have penalties rather than a tabu list, so in this paper, our aspiration criterion means ignoring the penalties (lines 3 & 4 in Figure 3 below, otherwise it is the same as the standard local search, pseudo code in Figure 2), if there is a move which can produce a new best solution. We shall call such a move, an *aspiration move*.

LocalSearchQAPAspiration(π,π^*, *g*, *h*)
{
1. sideways_count = 0
2. ***while****(there is a downwards move w.r.t.* $h(\pi)$ *or*
(there is a sideways move w.r.t. $h(\pi)$ *and sideways_count < 2)*
and termination criteria is not met)
{
//NOTE: the first term with the original objective function
//is the aspiration criteria, the second is the standard
//GLS, minimizing the augmented objective function
3. ***if*** *there exists a move, such that* $g(\pi) + \Delta g(\pi$ *with* π_i *and* π_j *swapped) < best cost so far*
4. $\pi = \pi$ *with the elements* π_i *and* π_j *swapped such that* $\Delta g(\pi$ *with* π_I *and* π_j *swapped) is minimised (ties are broken randomly)*
5. ***else***
6. $\pi = \pi$ *with the elements* π_i *and* π_j *swapped such that* $\Delta h(\pi$ *with* $\pi_{i7.}$ *and* π_j *swapped) is minimised, ties are broken randomly*

7. ***if*** $(\Delta h(\pi) == 0)$
8. sideways_count = sideways_count + 1
9. ***else***
10. sideways_count = 0

11. If $(g(\pi) < g(\pi^*))\ \pi^* = \pi$
12. }
13. ***return*** π
}

Figure 3: Pseudo code for local search for the QAP with aspiration moves for use with GLS

We have found that aspiration moves improve the performance (in terms of %relative error, a measure of solution quality, see equation (13)) of GLS in terms of the average best found solution over a run (see Figure 4), particularly when large values of lambda are used.

$$\%relative\_error(Cost) = \frac{Cost - Best\_Known\_Cost}{Best\_Known\_Cost} * 100 \quad (13)$$

(where Best_Known_Cost is the best known cost (this can be found in [4]) for that problem

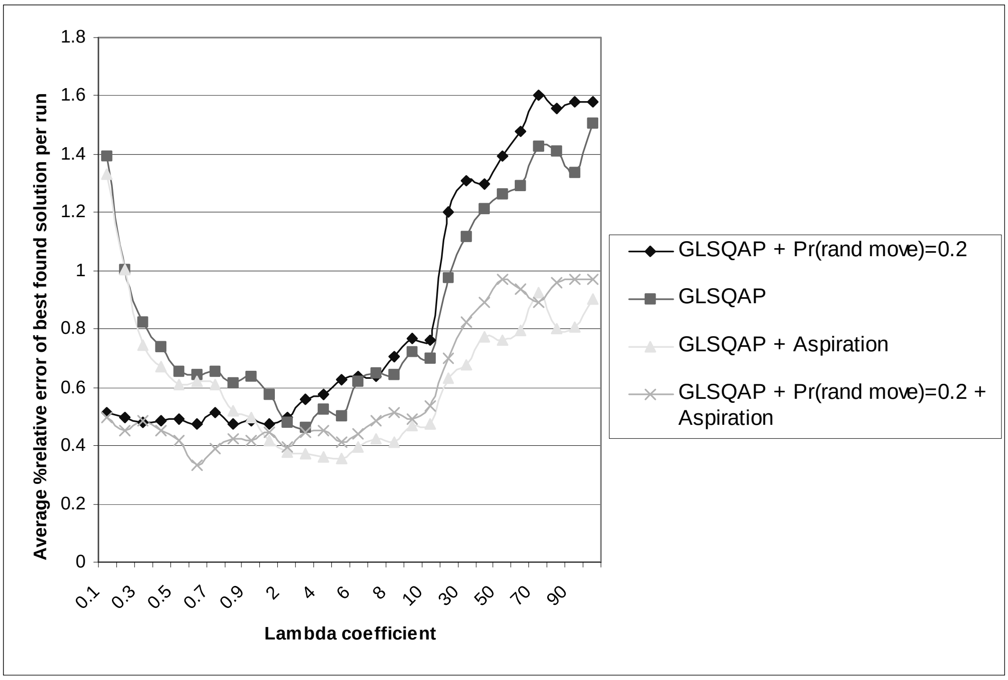

Figure 4: GLS variants over a range of lambda coefficients values on small to medium sized QAPLib [4] problems, average of 10 runs, 1000 * N repairs (N = problem size)

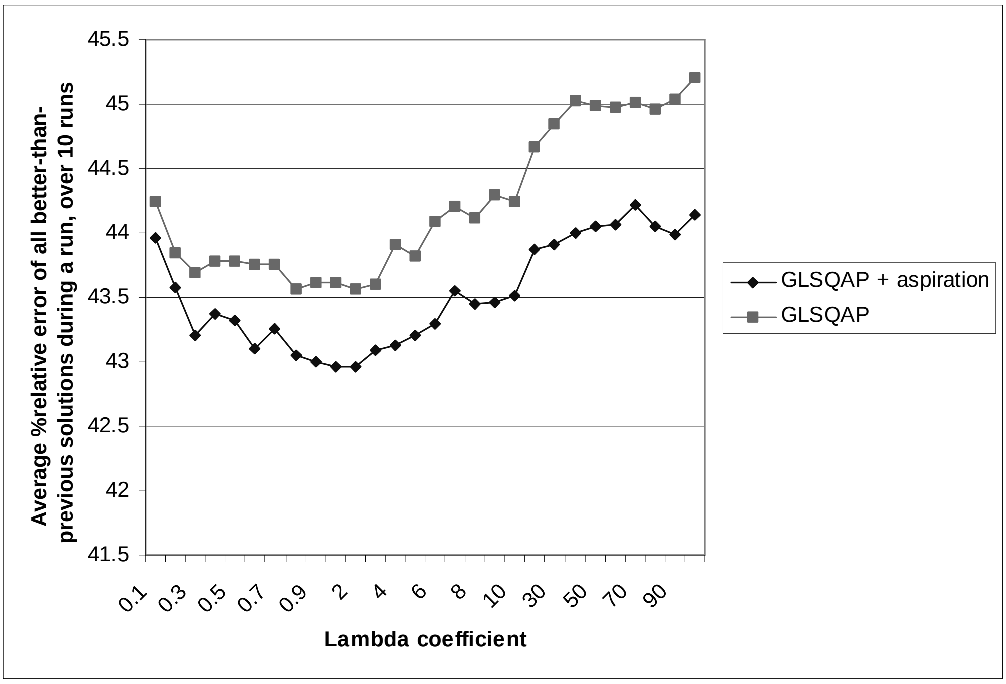

Figure 5: Average %relative error of better-than-previous solutions over all small-medium sized QAPLib problems, over 10 runs

We theorised that *aspiration moves work because they allow us to focus on minimising the original objective function at critical points during the search and also allows us to find more better-than-previous* *solutions[4] per run* (see Figure 6). This is particularly important when the lambda coefficient is large, as any penalty will have a larger effect on the local search algorithm, and is a plausible explanation for the improved performance of GLS when the lambda coefficient is large.

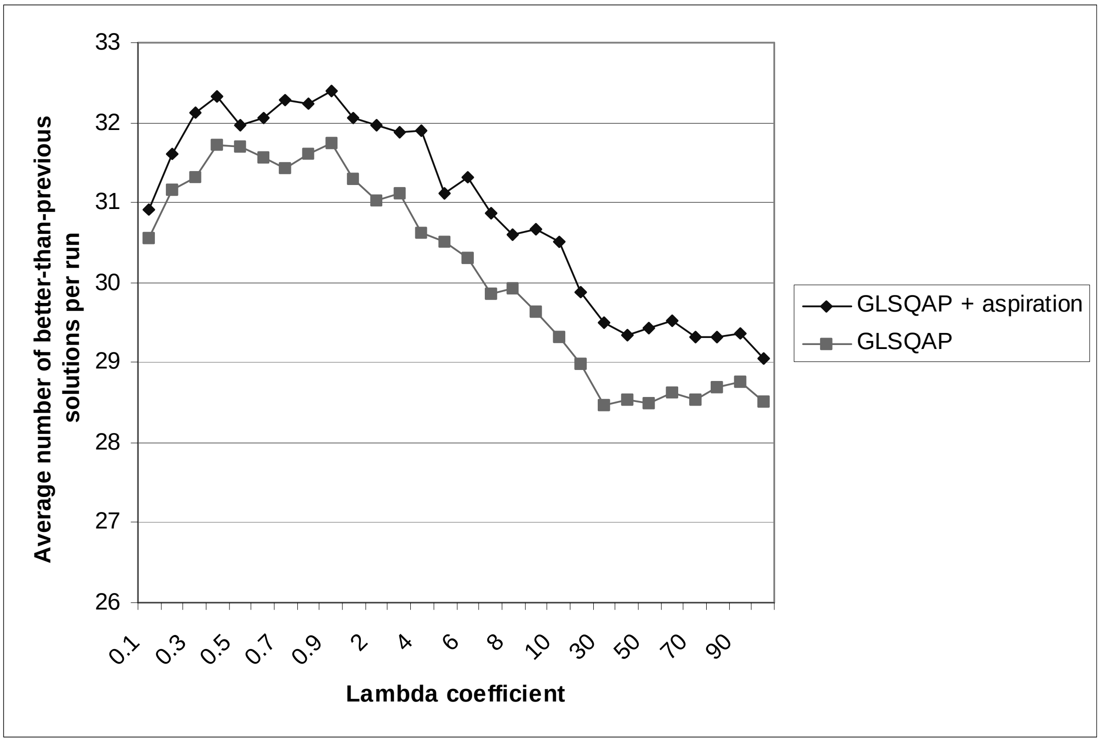

Figure 6: Average number of better-than-previous solution found per run on all small to medium sized QAPLib problems, over 10 runs

We ran controlled experiments, to try and substantiate this theory of why aspiration moves work. The first was to allow a GLS without aspiration to follow the standard GLS scheme p% of the time and 100-p% of the time choose a move according the original objective function. We found that simply allowing GLS to ignore penalties p% of the time, does not result in an increase in performance (for lack of space, we omit these results here), and so is not the sole reason for the success of aspiration moves.

So the reason that aspiration moves produced better results was not just because they allowed GLS to occasionally ignore the penalty term in the augmented objective function. During the runs of GLS with and without aspiration moves, we also recorded the average cost of each of the better-than-previous solutions over each run of GLS (see Figure 5). We found that when aspiration moves were used that this value was substantially lower than when aspiration moves were not used. We also found that GLS with aspiration moves found more better-than-previous solutions per run, than GLS without aspiration. This suggests that GLS with aspiration works as we theorised, because it allows GLS to find new best-found solutions that it might otherwise simply ignore due to penalties imposed on those solutions. We recorded several statistics about the quality of solutions visited during the search (including the average cost local minima visited, the average cost of solutions visited) and only the statistics on the quality of better-than-previous-found and best found solutions per run, varied between GLS with and without aspiration. This also suggested that it was precisely when and what aspiration does that is critical in its success.

[4] A better-than-previous solution, is one with a lower cost (in terms of the original objective function) than all the previous solutions visited so far, during a run

3.2 Adding random moves to GLS

The second experiment was to run GLS, allowing a random move to be chosen from the neighbourhood p% of the time, with the normal GLS scheme being followed the rest of the time (this technique was partly inspired by the walksat algorithm [20]). This was to check whether or not GLS was simply able to move into areas of the search space which would otherwise have been difficult to reach, due to penalties restricting GLS moves, when aspiration moves were added. These experiments gave rise to an increase in the performance of GLS when small values of lambda were used, although the increase in performance when the lambda coefficient was large that occurred with aspiration moves did not occur. In fact, from looking at the average entropy (see Figure 7, this is a measure of the spread, 0 would mean only one facility-location assignment was visited for a particular element in the permutation, 1 would mean all labels were facility-location assignments were present in the same quantities, see equation (14) for a definition of average entropy or [3] for the definition of entropy) of facility-location assignments, we observed that random moves had a completely different effect from aspiration moves in that they allowed GLS to diversify its search when lambda was too small, whereas GLSQAP with aspiration moves gave almost identical values to the basic GLSQAP.

$$Average\_entropy = \sum_{i=1}^{n}\sum_{j=1}^{n} -\frac{freq(\pi i = j)}{iterations} \cdot \frac{\log\left(\frac{freq(\pi i = j)}{iterations}\right)}{\log(n)} \tag{14}$$

where:

freq(πi=j) = the frequency of solutions visited where facility j is at location i,
iterations = the total number of solutions visited during the search &
n = the problem size in terms of the number of locations

3.3 Further studies of random moves

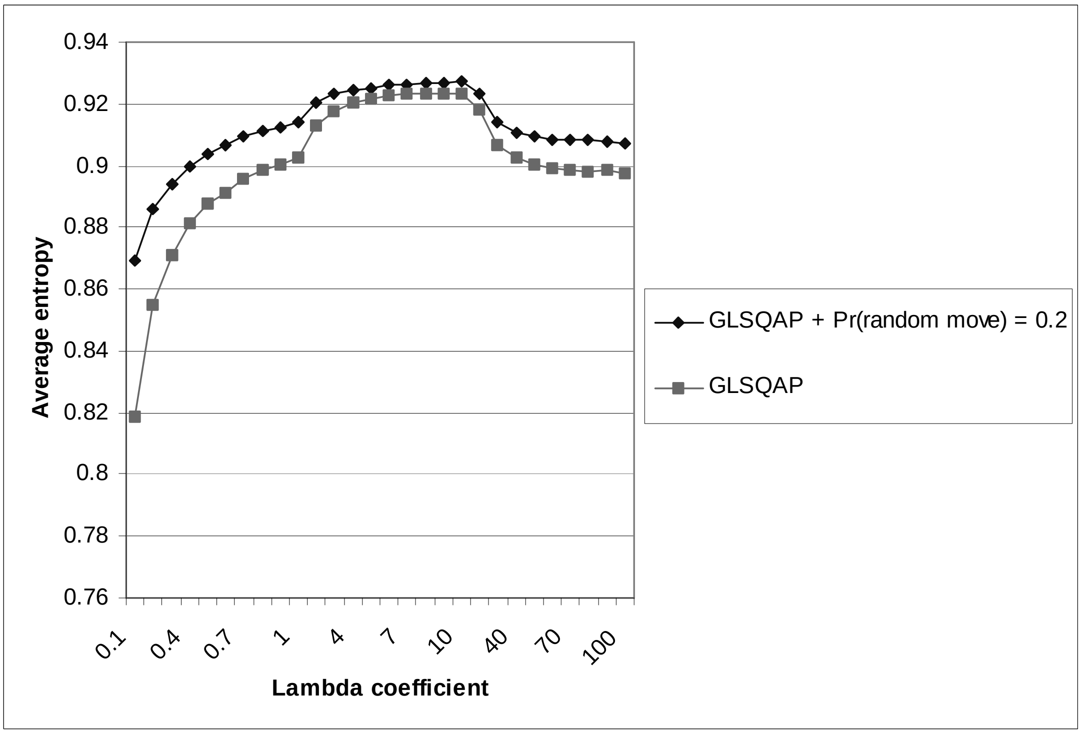

Figure 7: Average entropy of GLSQAP with random moves verses basic GLSQAP, over all small-medium sized QAPLib problems, over 10 runs

As already mentioned, whilst trying to understand more precisely why aspiration moves gave a performance improvement to GLS, we tried an additional scheme, whereby with probability p, we allowed GLS to make a move at random. We found that GLS without random moves, at low values of lambda produced a much less diverse search, than GLS with random moves, resulting in a better performance with respect to the best cost of solution of GLS with random moves. This suggests that the role of random moves is to help GLS move out of local minima, when GLS on it's own might not be able to do so. This is supported by the fact that the average entropy (see Figure 7) is higher when random moves are used with GLSQAP than GLSQAP without random moves. The number of repeated solutions (see Figure 8) is lower, for GLSQAP with random moves, when lambda is set to too low a value to allow escape from local minimas, although when lambda becomes larger, this value crosses over, so that GLSQAP with random moves produces more repeated solutions. This suggests that GLSQAP on it's own is slightly more efficient at escaping from local minima, when lambda is large enough, than random moves.

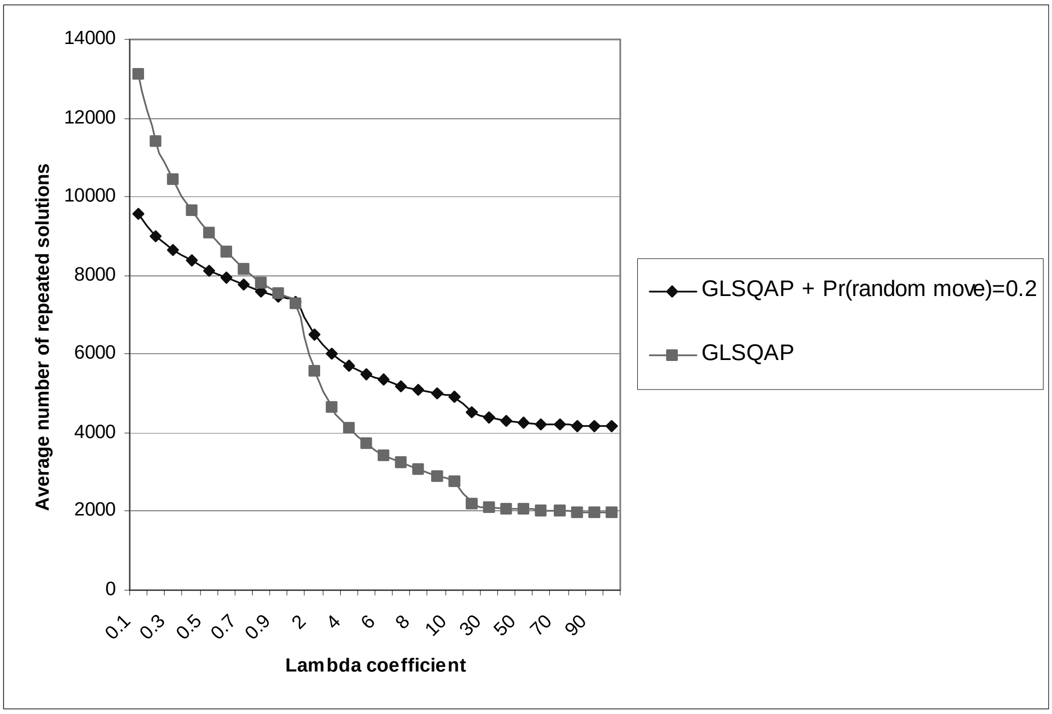

Figure 8: Average number of revisited solutions during runs of GLSQAP with and without random moves, over small-medium sized QAPLib problems, over 10 runs

4 Comparison with state of the art QAP algorithms

In this section, we compare our extended GLS against two state of the art QAP algorithms: Reactive Tabu Search [2] and Robust Tabu Search [24], but we should emphasise that this is only to show that our extended GLS has a place in solving the QAP, but not to show that GLS is by any means the "best" algorithm for the QAP.

We allowed each algorithm a maximum of $1000 \times N$ repairs (where N is the number of variables in the problem) and 10 runs each, taking the average deviation from the best known solution in every case and taking the average result of those 10 runs. We set GLS to use a lambda coefficient of 0.6, Pr(random move) = 0.2, and allowed GLS to make aspiration moves (we call this variant EGLS in the table). We also ran GLS without random moves and aspiration moves for comparison, with a lambda coefficient of 1 (this variant is called GLS in the table). The parameters for reactive tabu search (ReTS) and robust tabu search (RTS) were the standard parameters suggested in [2] and [24], although we used our own implementation (which according to our experience performs similarly to the original results in the papers). All algorithms were implemented on C++ and run on identical PCs running Linux. A summary of results are shown in Table 1.

These results show EGLS gives a comparable performance to both reactive tabu search and robust tabu search overall, and in some cases outperforms one or both (the bur* groups of problems and on the els19 problem) of them in terms of solution quality. In terms of CPU seconds, EGLS performs comparably with both reactive tabu search and robust tabu search. This is probably because all the algorithms use the same neighbourhood structure and updating of the objective function values, thus giving similar CPU times.

5 Conclusion

In this paper, we have presented an Extended Guided Local Search algorithm and its application to the Quadratic Assignment Problem. We have shown how two simple extensions of Guided Local Search, can dramatically increase the range of parameters under which GLS performs well. We have also studied and provided evidence on why they work. Since Guided Local Search is a general meta-heuristic and given our understanding of the extensions, we believe they should also generalise to other problems similar in nature to the QAP. Finally, we have shown that Guided Local Search with these two extensions gives comparable results to reactive tabu search and robust tabu search (two of the most famous heuristic methods for solving the Quadratic Assignment Problem), in some cases outperforming them, given the same number of iterations for each algorithm. Summarising, our Extended Guided Local Search algorithm, used with the parameters given in Section 4 of this paper, is a useful algorithm for solving QAP instances, as we have shown in the results of this paper.

	Mean %Relative error				Mean CPU seconds			
Problem group	GLS	EGLS	RTS	ReTS	GLS	EGLS	RTS	ReTS
bur*	0.001	**0.001**	0.002	0.084	4.5	4.5	4.5	4.3
chr*	2.350	1.988	1.516	1.909	1.4	1.3	1.2	1.1
els19	3.416	**0.000**	0.193	1.684	1.4	1.4	1.4	1.3
esc*	0.016	0.024	0.000	0.747	35.9	34.3	30.2	28.2
had*	0.000	0.000	0.000	0.008	0.9	0.9	0.9	0.8
kra*	0.631	0.605	0.105	0.213	5.3	5.3	5.3	4.9
lipa*	0.118	0.398	0.077	0.231	66.1	66.9	67.2	64.2
nug*	0.005	0.004	0.002	0.009	1.8	1.8	1.7	1.6
rou*	0.013	0.037	0.016	0.015	1.0	0.9	0.9	0.8
scr*	0.003	0.000	0.000	0.000	1.0	0.9	0.9	0.8
sko*	0.139	0.160	0.130	0.209	125.2	127.7	128.3	120.7
ste*	0.907	0.520	0.075	0.739	9.1	9.1	9.3	8.5
tai*a	0.271	0.811	0.680	0.430	31.7	31.5	31.2	29.3
tai*b	1.196	0.635	0.420	1.318	115.3	117.6	117.4	113.5
tai*c	1.347	0.063	0.039	0.022	626.4	583.9	553.6	532.7
tho*	0.132	0.180	0.141	0.221	256.5	260.6	255.0	244.1
wil*	0.093	0.100	0.087	0.146	109.6	112.1	113.1	106.3
Average	0.626	0.325	0.205	0.470	81.9	80.0	77.8	74.3

Table 1 : Summary of GLS verses reactive tabu search and robust tabu search: mean % relative error from best known solution and CPU seconds, over 10 runs, 1000 x N repairs per problem

6 Bibliography

1. Amin, A.: Simulated Jumping. Appears in Annals of Operations Research, 1998.
2. Battiti, R., Tecchiolli, G.: The Reactive Tabu Search. In ORSA Journal on Computing, 6(2):126-140, 1994.
3. Bishop, C.M., Hinton, G.: Neural Networks for Pattern Recognition. Clarendon Press 1995.

4. Burkard, R.E., Karisch, S.E., and Rendl, F.: QAPLIB - A Quadratic Assignment Problem Library. In Journal of Global Optimization 10:391-403, 1997.
5. Davenport A., Tsang E.P.K., Zhu, K., Wang C.J.: GENET: A Connectionist Architecture for Solving Constraint Satisfaction Problems by Iterative Improvement. In Proceedings of AAAI, 1994, p.325-330.
6. Davenport, A.: Extensions and Evaluation of GENET in Constraint Satisfaction. PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, July, 1997.
7. Glover, F.: Tabu search Part I. In Operations Research Society of America (ORSA) Journal on Computing, Vol. 1, 109-206, 1989.
8. Glover, F.: Tabu search Part II. In: Operations Research Society of America (ORSA) Journal on Computing, Vol. 2, 4-32, 1990.
9. Fleurent, C., Ferland, J.A.: Genetic Hybrids for the Quadratic Assignment Problem. In Pardalos, P., Wolkowicz, H., (eds.), Quadratic Assignment and Related Problems, Vol. 16, pages 173-187. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1994.
10. Kilby, P., Prosser, P., Shaw, P.: Guided Local Search for the Vehicle Routing Problem. In Proceedings of the 2nd International Conference on Metaheuristics, July 1997.
11. Lau, T.L., Tsang, E.P.K.: Applying a Mutation-Based Genetic Algorithm to Processor Configuration Problems. In Proceedings of 8th IEEE Conference on Tools with Artificial Intelligence (ICTAI'96), Toulouse, France, November 1996 .
12. Lau, T.L., Tsang, E.P.K.: Solving the Processor Configuration Problem with a Mutation-Based Genetic Algorithm. In International Journal on Artificial Intelligence Tools (IJAIT), World Scientific, Vol.6, No.4, 567-585, December 1997.
13. Lau, T.L., Tsang, E.P.K.: The Guided Genetic Algorithm and its application to the General Assignment Problems. In IEEE 10th International Conference on Tools with Artificial Intelligence (ICTAI'98), Taiwan, November 1998.
14. Lau, T.L., Tsang, E.P.K.: Solving Large Processor Configuration Problems with the Guided Genetic Algorithm. In IEEE 10th International Conference on Tools with Artificial Intelligence (ICTAI'98), Taiwan, November 1998.
15. Lau, T.L., Tsang, E.P.K.: Solving the Radio Link Frequency Assignment Problem with the Guided Genetic Algorithm. In Proceedings, NATO Symposium on Radio Length Frequency Assignment, Sharing and Conservation Systems (Aerospace), Aalborg, Demark, October 1998, Paper 14b.
16. Lau, T.L.: Guided Genetic Algorithm. PhD Thesis, Department of Computer Science, University of Essex, 1999.
17. Lau, T.L., Tsang, E.P.K.: Guided Genetic Algorithm and its application to Radio Link Frequency Assignment Problems. To appear in Journal of Constraints.
18. Mills, P., Tsang, E.P.K.: Guided Local Search for solving SAT and Weighted MAX-SAT Problems. In Journal of Automatic Reasoning, Special Issue on Satisfiability Problems, Kluwer, Vol.24, 2000, 205-223.
19. Pardalos,P.M., Rendl,F., Wolkowicz, H.: The Quadratic Assignment Problem: A Survey of Recent Developments. In P. Pardalos and H. Wolkowicz, editors, Quadratic Assignment and Related Problems, volume 16, pages 1-42. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1994.
20. Selman, B., Kautz, H., Cohen, B.: Noise Strategies for Improving Local Search. In Proceedings AAAI-94. 1994.
21. Stroustrup, B.: The C++ Programming Language - 3rd Edition. Addison Wesley 1997.
22. Stutzle T.: MAX-MIN Ant System for Quadratic Assignment Problems. Research Report AIDA-97-04, Department of Computer Schience, Darmstadt University of Technology, Germany, 1997.
23. Taillard, E.D.: Robust Tabu Search for the Quadratic Assignment Problem. In Parallel Computing, 17:443-455, 1991.
24. Taillard, E.D.: Comparison of Iterative Searches for the Quadratic Assignment Problem. Location Science, 1994.
25. Taillard, E.D., Gambardella, L.M.: Adaptive Memories for the Quadratic Assignment Problem. Research Report, IDSIA Lugano, Switzerland, 1997.
26. Thonemann, U.W., Bolte, A.: 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, 1994.

27.Tsang, E.P.K., Wang, C.J.: A Generic Neural Network Approach for Constraint Satisfaction Problems. In Taylor, J.G. (ed.), Neural network applications, Springer-Verlag, p.12-22, 1992.
28.Tsang, E.P.K., Voudouris, C.: Fast Local Search and Guided Local Search and their application to British Telecom's Workforce Scheduling Problem. In Operations Research Letters, Elsevier Science Publishers, Amsterdam, Vol.20, No.3, 119-127, March 1997.
29.Voudouris, C.: Guided Local Search for Combinatorial Optimisation Problems. Ph.D. thesis, Department of Computer Science, University of Essex. 1997.
30.Voudouris, C., Tsang, E.P.K.: Solving the Radio Link Frequency Assignment Problem using Guided Local Search. In Proceedings of NATO Symposium on Radio Length Frequency Assignment, Sharing and Conservation Systems (Aerospace), Aalborg, Demark, October 1998, Paper 14a.
31.Voudouris, C.: Guided Local Search - An Illustrative Example in Function Optimisation. In BT Technology Journal, Vol.16, No.3, July 1998, 46-50.
32.Voudouris, C., Tsang, E.P.K.: Guided Local Search and its application to the Travelling Salesman Problem. In European Journal of Operational Research, Anbar Publishing, Vol.113, Issue 2, March 1999, 469-499.
33.Wilhelm, M.R., Ward, T.L.: Solving Quadratic Assignment Problems by Simulated Annealing. In IIE Transaction, 19/1:107-119, 1987.