Parallel Computing 17 (1991) 443–455
North-Holland

Robust taboo search for the quadratic assignment problem

E. Taillard

École Polytechnique Fédérale de Lausanne, Département de Mathématiques, CH-1015 Lausanne, Switzerland

Received 12 July 1990
Final revision 25 July 1990

Abstract

Taillard, E., Robust taboo search for the quadratic assignment problem, Parallel Computing 17 (1991) 443–455.

An adaptation of taboo search to the quadratic assignment problem is discussed in this paper. This adaptation is efficient and robust, requiring less complexity and fewer parameters than earlier adaptations. In order to improve the speed of our taboo search, two parallelization methods are proposed and their efficiencies shown for a number of processors proportional to the size of the problem.

The best published solutions to many of the biggest problems have been improved and every previously best solution (probably optimal) of smaller problems has been found.

In addition, an easy way of generating random problems is proposed and good solutions of these problems, whose sizes are between 5 and 100, are given.

Keywords. Combinatorial optimization; taboo search; parallel algorithms; quadratic assignment problem; efficiency.

1. Introduction

For more than 30 years researchers have been proposing exact and heuristic methods for the quadratic assignment problem (QAP). The problem is so hard that an instance of size 36 proposed in 1961 by Steinberg [18] has not been solved yet. More precisely, exact methods need an amount of work that grows exponentially with the size of the problem and cannot be applied effectively beyond a size of 20, as noted by Burkard [2]. For bigger sizes, heuristic methods must be used. Two heuristics have performed particularly well for this problem: simulated annealing (SA) first adapted to QAP by Burkard et al. [3], and taboo search (TS) implemented by Skorin–Kapov [16,17].

Although obtaining the best results reported in the literature, TS has been applied in this setting by performing multiple solution passes for each problem, starting from different initial solutions and modifying parameters between the solutions. The goal of this paper is to propose a more robust form of TS for QAP that has a minimum number of parameters and is very easy to implement, capable of obtaining solutions comparable to the best previously found without requiring multiple passes and altered parameter values. Our method is efficient and succeeds in obtaining sub-optimal solutions (conjectured but not proved to be optimal) for problems up to size 64.

The paper is organized as follows: Section 2 presents a formulation of QAP and a summary of practical methods for solving this problem. Section 3 gives specific ingredients we used for QAP and, in particular justifies the values of the parameters we chose for our TS procedure.

The lack of homogeneity of the published problems led us to propose a simple method for generating problems randomly, that allows larger (and hence harder) problems to be created and tested by our TS procedure than those appearing in the literature. This is done in Section 4.

In the next section we propose two parallelization methods for TS. In particular we show experimentally that a number of processors proportional to the size of the problem can be efficiently used.

Finally, Section 6 reports numerical results obtained with our TS implementation.

2. The quadratic assignment problem

Mathematically, a concise way of presenting QAP is the following:

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} b_{\phi(i)\phi(j)} \qquad \phi\text{: permutation of } \{1 \dots N\}.$$

Practically, N units have to be assigned to N different locations, knowing that the distance between the locations i and j is a_{ij} and that a flow having a value of b_{rs} goes from unit r to unit s, such that the sum of products distance $\times$ flow is minimized. Examples of problems that may naturally be expressed in terms of QAP include:

- the placement of logical modules in a chip such that the total length of the connections is minimized;
- the distribution of medical services in a large hospital centre.

Many other examples exist and the reader is referred to the work of Finke et al. [8] for a recent overview of this subject.

The QAP contains the travelling salesman problem as a special case and hence is NP-hard. Moreover, no polynomial algorithm exists with a relative guaranteed error lower than a fixed constant for every instance of QAP unless P = NP, as demonstrated by Sahni et al. [15]. However, the asymptotic behaviour of randomly generated problems, under some assumptions, shows that the relative difference between the best and the worst solutions tends to zero if the problem size tends to infinity.

Table 1 compares the value of the solutions obtained by some methods; first, the method proposed by Nugent et al. [13], then the one of Elshafei [6]. The first results obtained using SA are the best published either in the original paper of Burkard et al. [3] or those obtained by Skorin–Kapov [16] using the SA code of Burkard et al.; the second results obtained using SA are those published in Connolly [5].

The results obtained using TS originate first from the first paper of Skorin–Kapov (Tabu-Navigation), second form her second paper where some extensions of TS (target analysis) for the QAP are discussed and third from the connectionnist approaches of Chakrapani et al. [4]. We have also reported the best solution known up to now and obtained by the implementation of TS of the present paper.

The best heuristic methods have been shown to provide very good solutions for sizes up to 40, which motivates the examination of problems of larger size. Accordingly, we include consideration of problems up to a size of 100 in this study, making use of the procedure we have designed for randomly generating problems of arbitrary size.

The fact that earlier versions of the TS procedure have obtained the best known results for QAP, both in terms of solution quality and CPU times, underlies our motivation to focus on this method and to seek a more robust and effective implementation.

3. Taboo search

The main ideas of TS may be briefly sketched as follows. The first ingredient (common to most heuristic and algorithmic procedures) is to define a *neighbourhood*, or a set of *moves* that may be applied to a given solution to produce a new one.

Among all the neighbouring solutions, TS seeks one with a best heuristic evaluation. In the simplest case such an evaluation dictates the choice of a move that improves most the objective function. If there are no improving moves (indicating a kind of local optimum), TS chooses one that least degrades the objective function. In order to avoid returning to the local optimum just visited, the reverse move now must be forbidden. This is done by storing this move – or more precisely a characterization of this move – in a data structure often managed like a circular list and called *taboo list*. This list contains a number s of elements defining forbidden (taboo) moves; the parameter s is called the *taboo list size*. However, the taboo list may forbid certain relevant or interesting moves, as exemplified by those that lead to a better solution than the best one found so far. Consequently, an *aspiration criterion* is introduced to allow taboo moves to be chosen if they are judged to be interesting.

A complete version of TS contains additional elements (see e.g. Glover [10,11]), but we will show how these basic notions alone can be used to construct an effective implementation for QAP.

3.1. Moves

A natural type of move that is very well suited for this problem consists of exchanging two units so that each occupies the location formerly occupied by the other. Starting from a

Table 1
Best solutions obtaind by some methods

Problem	Size	Method							
		Nugent	Elshafei	SA [a]	SA [b]	TS [c]	TS [d]	TS [e]	TS [f]
Nugent	15	1206	1228	1150	1150	1150	1150	1150	1150
	20	2678	2598	2570	2570	2570	2570	2570	2570
	30	6379	6250	6148	6124	6124	6194	6124	6124
Krarup	30	–	–	88900	–	88900	92210	88900	88900
	30	–	–	91420	–	91420	93520	91420	91420
Steinberg	36	–	–	9572	–	9526	9654	9526	9526
	36	–	–	15852	–	15852	16978	15852	15852
Skorin-Kapov	42	–	–	15956	–	15864	15864	15818	15812
	49	–	–	23662	–	23536	23472	23398	23386
	56	–	–	34916	–	34736	34788	34658	34458
	64	–	–	49020	–	48964	48928	48760	48498
	72	–	–	66924	–	66756	66794	66268	66256
	81	–	–	91432	–	91778	91770	91362	91008
	90	–	–	116460	–	116360	116258	116116	115534
Wilhelm and Ward	50	–	–	–	48816	–	–	–	48816
	100	–	–	–	273400	–	–	–	273044

[a] Best solution found with SA implemented by Burkard et al. [3].
[b] Best solution found with SA implemented by Connolly [5].
[c] Best solution found with TS, published in Skorin-Kapov [16] (using either Tabu-Navigation or a long term memory).
[d] Best solution found with TS, using target analysis (Skorin-Kapov [17]).
[e] Best solution found with a simulation of a Boltzmann machine that uses TS (Chakrapani et al. [4]).
[f] Best known solution (found with TS, using the implementation of the present paper).

placement ϕ, a neighbouring placement π is obtained by permuting units r and s:

$$\pi(k) = \phi(k) \ \forall k \neq r,s$$
$$\pi(r) = \phi(s)$$
$$\pi(s) = \phi(r).$$

If the matrices are symmetrical and with a null-diagonal (as it is the case for 'classical' examples) the value of a move, written $\Delta(\phi, r, s)$ is given by:

$$\Delta(\phi, r, s) = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(a_{ij} b_{\phi(i)\phi(j)} - a_{ij} b_{\pi(i)\pi(j)} \right)$$
$$= 2 \cdot \sum_{k \neq r,s} (a_{sk} - a_{rk})(b_{\phi(s)\phi(k)} - b_{\phi(r)\phi(k)}). \quad (1)$$

If the matrices are not symmetrical and/or without a null-diagonal, the expression – still computable in $O(N)$ – is a bit more complicated and is given by Burkard et al. [3]. Moreover, if the exchange of units r and s in a placement ϕ provides a placement π, it is possible (for u and v different from r or s) to compute the value $\Delta(\pi, u, v)$ in a constant time if one takes care to store the value of $\Delta(\phi, u, v)$ at the previous step of taboo search:

$$\Delta(\pi, u, v) = \Delta(\phi, u, v)$$
$$+ 2(a_{ru} - a_{rv} + a_{sv} - a_{su})(b_{\pi(s)\pi(u)} - b_{\pi(s)\pi(v)} + b_{\pi(r)\pi(v)} - b_{\pi(r)\pi(u)}). \quad (2)$$

If u or v is equal to r or s, it is possible to compute the value of $\Delta(\pi, u, v)$ by the use of the expression (1). Consequently, the complete evaluation of the neighbourhood takes $O(N^2)$ operations, as shown in Frieze et al. [9].

A fast evaluation of the quality of the moves is an important factor contributing to the efficiency of search. This evaluation may be exact as in the travelling salesman application of Fiechter [7] or in the flow shop sequencing of Taillard [19] or approximate as in the job shop scheduling of Taillard [20]. Moreover, the moves must have good properties in relation to the problem considered. It would be inappropriate for QAP to employ a move that inserts a unit r adjacent to a unit s in a permutation ϕ, because that would change the location of every unit between r and s in the permutation – a real disaster!

3.2. Taboo list

We have now to define the type of the elements of the taboo list. Skorin–Kapov [16] has proposed to forbid the exchange of two units that were exchanged during the preceding s iterations. Formally, the taboo list is constituted of pairs (i, j) of units that cannot be exchanged. The prohibition of the list may be implemented with a 2-dimensional array of integers where the element (i, j) identifies the value of the future iteration at which these two units may again be exchanged with each other. By this means, testing whether a move is taboo or not requires only one comparison. It is important to evaluate a taboo condition in a constant time in order to evaluate the neighbourhood in a time proportional to its size, and hence to avoid a growth in the complexity of the search.

The implementation of Skorin–Kapov does not use formula (2) to accelerate the evaluation of moves, which reinforces her conclusion of the superiority of TS to SA (as implemented by Burkard et al. [3]).

In his diploma project, Rogger [14] has tried many types of taboo lists and we chose the one that proves most convenient if the size s of the list is changed, because we employ a rule that

varies the value of this parameter (advantages of such a list in general setting are described by Glover [11]). Specifically, our taboo list is constructed as follows: for every unit and location, the latest iteration at which the unit occupied that location is recorded. A move is taboo if it assigns both interchanged units to locations they had occupied within the s most recent iterations. This type of list provides results that are comparable to those obtained with the type proposed by Skorin–Kapov when their 'optimal' sizes are given, but is less sensitive to the parameter s.

3.3. Aspiration function

For every problem we use the classical aspiration function that allows a taboo move to be selected if it leads to a solution better than the best found so far. However, the best known solutions to some problems could not always be obtained with this minimal implementation – i.e. by using only one type of move, one taboo list and the trivial aspiration function. Closer examination of the problems where the implementation performed more poorly (in particular the problem proposed by Elshafei [6]), disclosed what was conceptually wrong: these problems have a very special flow matrix containing a lot of null or very small values and some very high flows. If the search at some point badly locates the units with high flows at a local optimum, then TS will perform moves with small costs because the first exchange of the crucial units would greatly increase the value of the objective function.

The trick we propose to overcome this difficulty is very simple: for the problems with very heterogeneous flows or costs, a move will pass the aspiration criterion and be selected, with solution quality an entirely subordinate measure, if the exchange will place both units at locations they have not occupied during the last t iterations. Conceptually, this function can be viewed as a longer term diversification process that is activated to forbid moves that fail to satisfy its condition whenever some pair of units meets the criterion successfully.

3.4. Taboo list size

The choice of taboo list size is critical; if its value is too small, cycling may occur in the search process while if its value is too large, appealing moves may be forbidden and leading to the exploration of lower quality solutions, producing a larger number of iterations to find the solution desired.

However, it is false to think that the cycling phenomenon invariably appears as a function of a small taboo list size, because we observed, for the problem of size 15 of Nugent et al. [13], a cycling with a taboo list size set to $s = 30$ and this cycling was not observed for sizes between 26 and 29. To overcome the problem related to the search of the optimal taboo list size, we propose the following trick: rather than fixing the size s at a constant value during the whole search, we propose to change this value randomly during the search. Practically, s will be chosen between $s_{\min}$ and $s_{\max} = s_{\min} + \Delta$ and frequently changed, for example every $2 \cdot s_{\max}$ iterations (in order to have some probability to perform some iterations with $s = s_{\max}$).

In *Fig. 1*, we are interested in the practical values that have to be given to $s_{\min}$ and Δ to solve (optimally) the Nugent's problem with $N = 15$. When we succeed in finding an optimal solution 30 times, starting with independent initial solutions and using less than 10 000 iterations, we plot by a black square in the figure. It must be noted that the allowed number of iterations is about 20 times the mean number of iterations needed to solve this problem when $s_{\min}$ and Δ are set to their 'optimal' values.

In *Fig. 1*, we first note that the introduction of a random taboo list size ($\Delta > 0$) makes the search much more reliable. Second, there is a very wide range that may practically be given to $s_{\min}$ and Δ.

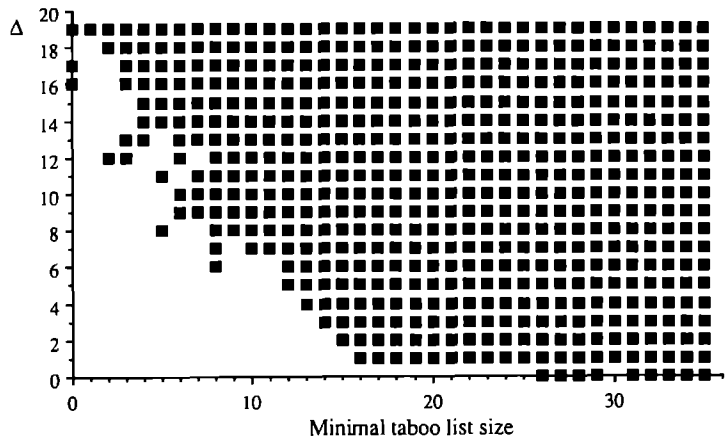

Fig. 1. Feasible values of taboo list size for Nugent's problem ($N = 15$).

In fact, for most problems we found the best solutions known up to now using $s_{min} = \lfloor 0.9\ N \rfloor$ and $s_{max} = \lceil 1.1\ N \rceil$. However, these solutions are sometimes found more rapidly with a slightly larger taboo list size for small problems ($N \leqslant 15$) and for problems with non-regular matrices (Elshafei, Steinberg) or for big problems (Skorin–Kapov, Krarup) where the second aspiration function is used, the taboo list size has to be reduced, approximately 30%. The parameter t of the second aspiration function depends strongly on the problem. For the most irregular one (Elshafei) t was set to 400 and for bigger problems it was set between 3000 and 10000.

In *Fig. 2* we are interested in the mean number of iterations to find a suboptimal solution, for the first instance of Steinberg's problem. So we try to 'solve' this problem 30 times, starting with independent initial solutions and allowing at most 500000 iterations. This is done for 3 values of Δ (0, 5 and 10) and 7 values of s_{min} (0, 5, ..., 30).

When the taboo list size is fixed ($\Delta = 0$), it is important to choose it adequately: for our example, we observe a cycling for sizes inferior to 10, while the optimal size is about 20, and the mean number of iterations increase of about 50% for a size of 30. But the introduction of randomness ($\Delta = 5$ or $\Delta = 10$) makes it possible to find the searched solution in a more reliable way (the interval of good values of s_{min} is bigger) and one needs less iterations (-30%) than for a fixed size.

It must be noted that the regularity of the curve of *Fig. 2* is generally not reproducible for every problem. There are often local minima at various places. This is due, as we are going to

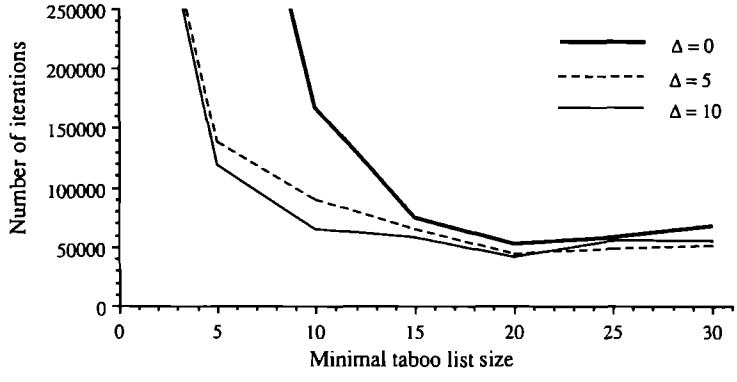

Fig. 2. Solving Steinberg's problem sub-optimally.

see in Section 5, to the number of iterations needed to find a sub-optimal solution: this number is a random variable (the initial solution is chosen at random) from an exponential distribution. As the standard deviation of this distribution is equal to the mean, the estimation of the mean is very imprecise. Consequently we are interested in the mean of the *logarithm* (in base 10) of the number of iterations. This measure seems to be an artificial one, but it is more precise since the standard deviation is less than 0.7 for every problem considered.

4. Random problems

The problems published in the literature have a major disadvantage: duplicating and using the data may be hard because the matrices are recorded in extensive tables, and sometimes printed in a unreadable way.

Accordingly, we propose a simple way to generate problems automatically; these problems will be generally more difficult to solve (with our TS) than the published ones because the largest problems whose sub-optimal solutions can be routinely verified have a size of 35 to 40 and we have most probably solved optimally the problems of Steinberg ($N = 36$) and Skorin–Kapov ($N = 42$ to $N = 64$).

In our generating procedure, the coefficients of the distance and flow matrices are integers, randomly and uniformly generated between 0 and 99 (contained). The random number generator, analyzed in the book of Bratley et al. [1], is based on the recursive formula:

$$X_k = (aX_{k-1}) \text{Mod } m,$$

where $a = 16807$, $m = 2^{31} - 1$ and $X_0 = 123456789$.

The generating procedure may be written in pseudo-code as follows:

```
k := 0
For i=1 to N do
    a_ii=0
    b_ii := 0
End for
For i=1 to N-1 do
    For j=i+1 to N do
        k := k+1
        a_ij := |(100 X_k) / m|
        a_ji := a_ij
    End for
End for
For i=1 to N-1 do
    For j=1 to N do
        k := k+1
        b_ij := |(100 X_k) / m|
        b_ji := b_ij
    End for
End for
```

This procedure may be implemented as written here, but the integers must be coded on 64 bits. It would be very easy to generate other problems by changing the initial value of X_0.

5. Parallel taboo search

Without changing the sequential algorithm significantly, the most efficient way of executing a TS concurrently is to distribute on many processors the component that requires the most

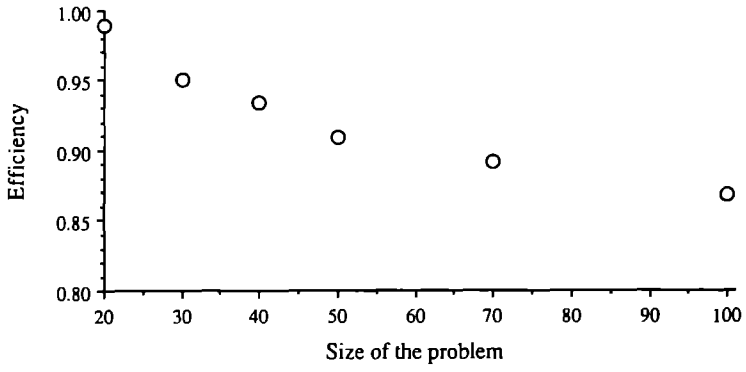

Fig. 3. Parallel efficiency of TS.

CPU time. For QAP, this component consists of evaluating the neighbourhood of currently available moves, that is to say, the values $\Delta(\phi, \cdot, \cdot)$. The parallelization idea is to divide the neighbourhood into p parts having about the same size, and to evaluate each of these parts on p different processors. In order to balance the work between the processors, the partition of the neighbourhood must be done carefully since there are $O(N)$ moves that need a computation time of $O(N)$ each, using formula (1), and since $O(N^2)$ moves need a constant computation time using formula (2), both types of moves must be divided into p parts, such that every processor has about the same amount of work.

Having done that, every processor computes the values of the moves that are attributed to it. Then, it communicates to the others its proposed move (the best move it found) and receives $p-1$ moves proposed from the other processors. Every processor chooses and performs the best move proposed to (or by) it, updating ϕ and the new value of $\Delta(\phi,\cdot,\cdot)$. If there are many proposed moves that are of the same quality, the processors will retain, for example, only the move that was proposed by the processor that has not had an accepted proposition for the greatest number of iterations.

This parallelization method is very general and provides good speed-up results (see Fiechter [7] and Taillard [19]). In *Fig. 3*, we are interested in the efficiency of such a parallelization method for QAP. So we have implemented our TS on a ring of transputers; the number of processors of the ring was proportional to the size of the problem (precisely $N/10$).

We remark that the efficiency (the division of the theoretical CPU time with an ideal speed-up by the CPU time effectively observed) decreases slightly as N increase, but an efficiency of more than 85% is possible with 10 processors.

Another way of parallelizing consists in performing many independent searches, starting every one with different initial solutions. This technique is widely used in sequential computation by the repeated execution of an algorithm (which could be TS or SA or any other method), modifying the initial solution after each execution. These algorithms may be seen as procedures that provide the desired solution with a probability $p(t)$ if they run during a time t. But, if $p(t) = 1 - e^{-\lambda t}$, with $\lambda > 0$, it is perfectly equivalent (in terms of probability) to perform one search during a time t or p searches during a time p/t.

For many problems solved by TS (graph colouring (Mohr [12]), scheduling (Taillard [19,20])) the empirical function $p(t)$ approximates an exponential distribution. This trivially means that TS implemented only with a short term memory (one taboo list, always active) and with no long term memory, has no advantage in being restarted.

For QAP, we observe again the same behaviour, as shown in *Fig. 4* where the empirical probability is plotted as a function of the number of iterations. (This is equivalent to $p(t)$ because every iteration takes a constant time for a given size of problem.) In order to evaluate

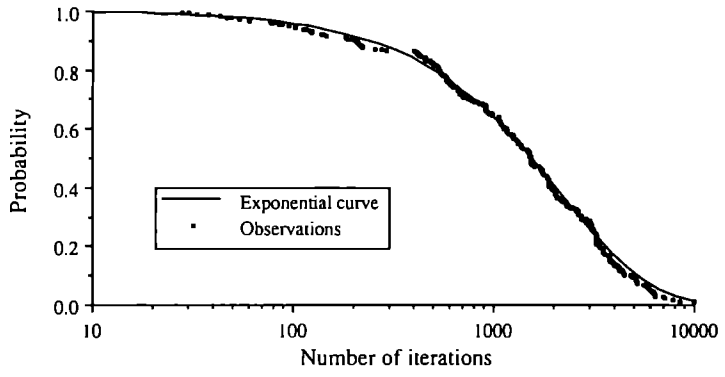

Fig. 4. Probability of not having found the sub-optimal solution ($N = 15$).

this function, we have solved our randomly generated problem of size fifteen 300 times to obtain the best known solution. We have then superimposed on the observed points the best exponential curve (in the sense of a least mean squares fit).

We see that the empirical observations approximate an exponential curve, which means that almost ideal speed-ups may be obtained by concurrent searches.

However, a minimal number of iterations have to be performed – let us say a number proportional to N, corresponding to the number of permutations that a descent method performs.

Finally, let us quote a quite different parallelization method, based on Boltzmann machine, that was successfully implemented by Chakrapani et al. [4]. It is shown in this paper that it is better to perform a TS process on the nodes of the Boltzmann machine rather than to perform a usual SA process.

6. Numerical results

First, we show the quality of our TS approach by comparing in *Table 2* some algorithms of diverse complexity: first the arbitrary choice of a solution, whose complexity, including the computation of the objective function, is $O(N^2)$; second, the exchange algorithm, that stops when the first local optimum is reached – empirically in $O(N^3)$, see the work of Frieze et al. [9] – and then our TS when 1000, $4N$ and N^2 iterations are performed – having respectively a complexity of $O(N^3)$, $O(N^3)$ and $O(N^4)$. We choose such numbers of iterations first because Skorin–Kapov [16] gives her results for 1000 and $4N$ iterations of TS in her paper and second because we were interested in performing a number of iterations that grows faster than the size of the problem.

In this table, we give the mean percentage above the best known solutions of these heuristic methods. Let us mention that we have always obtained the best known solutions by our TS, possibly allowing more iterations; in particular we improved the best known value of every problem proposed by Skorin–Kapov. The results are obtained by 'solving' every problem 30 times, starting with different initial solutions, the same 30 starting solution for every heuristic method.

In *Table 3*, we are interested in solving sub-optimally some of these problems. Again, we have started with 30 different solutions and we give in this table the following:

- The mean number of iterations needed to get the sub-optimum, but as the standard deviation of this measure is about the same order of magnitude, we give a more precise measure, that is

Table 2
Percents above best known solutions provided by some heuristic methods

Source	Size (N)	Random solution	First local min.	TS 1000 steps	TS $4N$ steps	TS N^2 steps	Best known solution
Random	5	36.1 *	8.9 *	0.0 *	0.0 *	0.0 *	12902 [c]
($X_0 = 123456789$)	6	16.2 *	6.7 *	0.0 *	0.2 *	0.0 *	29432 [c]
	7	18.1	4.5 *	0.0 *	2.7 *	0.5 *	53976 [c]
	8	21.3	8.6 *	0.0 *	1.3 *	0.1 *	77502 [c]
	9	23.5	7.3 *	0.0 *	0.7 *	0.1 *	94622 [c]
	10	20.9	6.0	0.0 *	1.4 *	0.7 *	135028 [c]
	12	26.3	7.7	0.6 *	3.5 *	1.1 *	224416 [c]
	15	16.8	4.0	0.1 *	1.6	0.7 *	388214 [c]
	17	19.2	4.8	0.6 *	2.2	1.1 *	491812 [c]
	20	19.1	5.4	0.9 *	2.7	1.5	703482 [c]
	25	17.5	4.8	1.3	2.8	1.5	1167256 [c]
	30	16.6	4.7	1.4	2.4	1.5	1818146 [c]
	35	17.4	4.5	1.6	2.5	1.5	2422002 [c]
	40	16.8	4.3	1.4	2.3	1.2	3146514
	50	16.6	4.3	1.8	2.4	1.5	4951186
	60	15.1	3.4	1.1	1.6	0.7	7272020
	80	13.8	3.2	1.4	1.6	1.0	13582038
	100	12.5	2.5	0.8	1.2	0.5	21245778
Nugent	5	15.5 *	6.9 *	0.0 *	0.4 *	0.1 *	50 [c]
	6	10.9 *	4.9 *	0.0 *	0.0 *	0.0 *	86 [c]
	7	12.4	3.7 *	0.0 *	0.1 *	0.0 *	148 [c]
	8	18.4	2.5 *	0.0 *	0.1 *	0.0 *	214 [c]
	12	24.2	4.0 *	0.0 *	1.5 *	0.9 *	578 [c]
	15	26.7	3.2	0.0 *	1.0 *	0.2 *	1150 [c]
	20	24.2	3.5	0.1 *	1.4 *	0.4 *	2570 [c]
	30	27.1	3.5	0.3 *	1.7 *	0.4 *	6124 [c]
Elshafei	19	122.6	44.5	3.3 [d] *	22.9 *	11.4 [d] *	17212548 [c]
Krarup [a]	30	43.8	7.3	2.9 *	4.5	3.0 *	88900 [c]
	30	42.1	4.9	1.0 *	2.9	1.1 *	91420 [c]
Steinberg [a]	36	106.7	11.9	4.1	6.5	3.6	9526 [c]
	36	320.7	25.5	9.2	16.4	8.5	15852 [c]
Skorin-Kapov [a]	42	23.5	2.9	0.6	1.5	0.4 *	15812 [c]
	49	21.3	2.9	0.6	1.7	0.2 *	23386 [c]
	56	21.4	2.9	1.0	2.0	0.5	34458 [c]
	64	19.2	2.5	0.7	1.5	0.4	48498 [c]
	72	18.2	2.5	0.9	1.5	0.4	66256
	81	17.7	2.1	0.8	1.2	0.4	91008
	90	17.1	2.0	0.9	1.3	0.4	115534
Wilhelm and Ward [b]	50	11.9	1.5	0.4	0.8	0.2	48816 [c]
	100	9.2	1.0	0.5	0.7	0.2	273044

* The best known solution was found once at most during the 30 trial.
[a] Data obtained via J. Skorin-Kapov (private communication).
[b] Data obtained via D.T. Connolly (privvate communication).
[c] Provably or probably optimal solution.
[d] The parameter t was set to $N^2/2$.

- the mean of the logarithm (in base 10) of the number of iterations and
- the standard deviation of this measure; we then give the parameters we used (not necessary optimal ones), these are:

Table 3
Amount of work needed to solve sub-optimally some problems

Source	Size (N)	Mean (iterations)	Mean log(iter.)	Std. dev. log(iter.)	s_{min}	s_{max}	t
Random	5	7.6	0.839	0.205	4	6	∞
($X_0 = 123456789$)	6	6.6	0.734	0.273	6	10	∞
	7	25.7	1.195	0.469	10	14	∞
	8	29.4	1.369	0.388	12	16	∞
	9	31.7	1.409	0.306	8	10	∞
	10	137.1	1.975	0.414	15	20	∞
	12	210.7	2.032	0.495	12	18	∞
	15	2168.0	3.099	0.447	15	19	∞
	17	5020.4	3.496	0.546	17	21	∞
	20	34279.0	4.373	0.464	18	22	∞
	25	80280.4	4.616	0.578	22	28	∞
	30	146315.7	4.967	0.451	27	33	∞
	30	70517.3	4.605	0.544	13	21	6000
	35	448514.5	5.388	0.593	17	29	7000
Nugent	5	10.9	0.856	0.397	4	6	∞
	6	8.3	0.869	0.223	5	7	∞
	7	14.5	1.047	0.341	6	8	∞
	8	13.8	1.058	0.272	7	9	∞
	12	231.5	2.211	0.411	10	14	∞
	15	752.6	2.584	0.617	13	17	∞
	20	1430.9	2.933	0.504	18	22	∞
	30	24712.8	4.070	0.711	27	33	∞
	30	14112.4	3.951	0.595	17	27	4000
Elshafei	19	4106.0	3.509	0.317	8	10	400
Krarup	30	23935.4	4.170	0.543	15	25	3000
	30	30251.3	4.357	0.373	15	25	3000
Steinberg	36	41312.7	4.506	0.361	20	30	4000
	36	11955.0	4.016	0.229	20	30	4000
Skorin-Kapov	42	26836.5	3.978	0.589	21	37	4000
	49	280662.6	5.257	0.517	25	43	8000
	56	537061.2	5.472	0.565	30	47	10000
	64	687651.9	5.468	0.632	38	55	10000
Wilhelm et al.	50	277409.6	5.191	0.488	31	40	8000

- the minimal size s_{min} of the taboo list and
- s_{max} its maximal size and finally
- the value of t, corresponding to the parameter of the second aspiration function (or ∞ if this last was not used).

Although it is possible to solve Nugent's and our random problem of size 30 without the second aspiration function, we undertook to solve it with this facility and we see that the introduction of a long term memory is useful for big problems. In general, the number of iterations grows very fast with the size of the problem and it becomes very expensive to solve problems having a size greater than 35 or 40.

Finally, let us mention that we used an implementation with 10 transputers (T800C-G20S) to get these results. In these conditions, the CPU time per iteration is given approximately by $6.2 \cdot N^2$ μs.

7. Conclusion

We show in this paper that it is possible to find very good solutions to QAP using a procedure based on TS and fixing its parameters a priori. In particular, if N^2 iteration are allowed and the taboo list size is varied randomly in a range of 10% about the size of the problem, it is possible to reach solutions of excellent quality, depending more on the type of problem than on its size, for $N \geqslant 20$.

The implementation of our TS, in addition to its simplicity, reduces the computational complexity of earlier TS implementations for QAP by a factor of N, and the complexity may be reduced by a factor N once more with the use of a number of processors proportional to N.

It is shown that the use of an elementary TS – one taboo list and the trivial aspiration function – works well for most problems up to a size of 30. For bigger problems, the use of another aspiration function is proposed – adding a new parameter but without changing the complexity (work and memory requirements) of the algorithm – and some problems up to the size of 64 may be solved sub-optimally. However, we think that the search of sub-optimal solutions for bigger problems must be done by another procedure, using for example more elaborate concepts of TS (target analysis or a more sophisticated long term memory).

Finally, a reliable way of generating random problems is given, that considerably reduces the work needed to get instances of them.

Acknowledgment

The author would like to thank Jadranka Skorin–Kapov for sending Krarup's, Steinberg's and her problems' data, David Connolly for sending Wilhelm et al. data, and Fred Glover who kindly brought, during his stay at the Swiss Federal Institute of Technology of Lausanne, his valuable and multiple suggestions to improve the presentation of this paper.

References

[1] P. Bratley, B.L. Fox and L.E. Schrage, *A guide to Simulation* (Springer, New York, 1983).
[2] R.E. Burkard, Quadratic assignment problems, *EJOR* 15 (1984) 283–289.
[3] R.E. Burkard and F. Rendl, A thermodynamically motivated simulation procedure for combinatorial optimization problems, *EJOR* 17 (1984) 169–174.
[4] J. Chakrapani and J. Skorin-Kapov, Connectionist approaches to the quadratic assignment problem, report HAR-90-09, W.A. Harriman School for Management and Policy, SUNY at Stony Brook, Stony Brook, NY, 11794–3775, 1990.
[5] D.T. Connolly, An improved annealing scheme for the QAP, *EJOR* 46 (1990) 93–100.
[6] A.E. Elshafei, Hospital layout as a quadratic assignment problem, *OR Q* 28 (1977) 167–179.
[7] C.-N. Fiechter, A parallel taboo search algorithm for large Travelling Salesman Problem, report ORWP 90/01, DMA, Swiss Federal Institute of Technology of Lausanne, CH-1015 Lausanne, Switzerland, 1990.
[8] G. Finke, R.E. Burkard and F. Rendl, Quadratic assignment problems, *Annals Discrete Math.* 31 (1987) 61–82.
[9] A.M. Frieze, J. Yadegar, S. El-Horbaty and D. Parkinson, Algorithms for assignment problems on an array processor, *Parallel Comput.* 11 (1989) 151–162.
[10] F. Glover, Tabu Search – part I, *ORSA J. Comput.* 1 (3) (1989) 190–206.
[11] F. Glover, Tabu search – part II, *ORSA J. Comput.* 2 (1) (1990) 4–32.
[12] Th. Mohr, Parallel tabu search algorithms for the graph coloring problem, report ORWP 88/11, DMA, Swiss Federal Institute of Technology of Lausanne, CH-1015 Lausanne, Switzerland, 1988.
[13] Ch.E. Nugent, Th.E. Vollmann and J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations, *Oper. Res.*. 16 (1968) 150–173.
[14] A. Rogger, Applications des techniques parallèles de recherche tabou au problème de l'affectation quadratique, diploma project, DMA, Swiss Federal Institute of Technology of Lausanne, CH-1015 Lausanne, Switzerland, 1989.

[15] S. Sahni and T. Gonzalez, P-complete approximation problems, *J. ACM* 23 (1976) 555–565.
[16] J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem, *ORSA J. Comput.* 2 (1) (1990) 33–45.
[17] J. Skorin-Kapov, Extension of a tabu search adaptation to the quadratic assignment problem, report HAR-90-006, W.A. Harriman School for Management and Policy, SUNY at Stony Brook, NY, 11794–3775, 1990.
[18] L. Steinberg, The backboard wiring problem: a placement algorithm, *SIAM Rev.* 3 (1) (1961) 37–50.
[19] E. Taillard, Some efficient heuristic methods for the flowshop sequencing problem, *EJOR* 47 (1) (1990) 65–74.
[20] E. Taillard, Parallel taboo search for the job shop scheduling problem, report ORWP 89/11, DMA, Swiss Federal Institute of Technology of Lausanne, CH-1015 Lausanne, Switzerland, 1989.