To appear in *Location Science* 3.

CRT-989

COMPARISON OF ITERATIVE SEARCHES FOR THE QUADRATIC ASSIGNMENT PROBLEM

Centre de recherche sur les transports
Université de Montréal
C. P. 6128, succursale Centre-Ville
Montréal, Canada H3C 3J7
e-mail : taillard@crt.umontreal.ca

Centre de recherche sur les transports — Publication N^{o} 989
Juillet 1994 (révision juillet 1995)

ÉCOLE PLOYTECHNIQUE FÉDÉRALE DE LAUSANNE
(EPFL)

DÉPARTEMENT DE MATHÉMATIQUES

ORWP 4/94

JUILLET 1994

Résumé :

Ce papier compare les méthodes heuristiques les plus efficaces pour le problème de l'affectation quadratique. Ces méthodes sont connues sous les noms de *recherche tabou stricte*, *recherche tabou robuste*, *recherche tabou réactive* et *algorithme génétique hybride*. Il est montré que l'efficacité de ces méthodes dépend fortement du type de problème traité et qu'aucune ne surpasse toutes les autres. Une technique rapide pour affiner les paramètres des mémoires à court terme des recherches tabou est proposée et sa validité est vérifiée expérimentalement sur des recherches de longue durée. Un nouveau type de problème d'affectation quadratique, survenant dans la conception de trames de gris pour images de synthèse, est proposé et il est montré comment les méthodes itératives existantes peuvent être adaptées et améliorées pour ce problème spécifique. Finalement, la manière usuelle d'implanter une approximation d'une recherche tabou stricte est discutée et de meilleures approximations sont proposées.

Mots clés : Affectation quadratique, recherche tabou, algorithmes génétiques.

COMPARISON OF ITERATIVE SEARCHES FOR THE QUADRATIC ASSIGNMENT PROBLEM

Éric D. Taillard*

Abstract :

This paper compares some of the most efficient heuristic methods for the quadratic assignment problem. These methods are known as *strict taboo search, robust taboo search, reactive taboo search* and *genetic hybrids*. It is shown that the efficiency of these methods strongly depends on the problem type and that no one method is better than all the others. A fast method for tuning the short term memory parameters of taboo searches is proposed and its validity is experimentally verified on long searches. A new type of quadratic assignment problem occurring in the design of grey patterns is proposed and it is shown how to adapt and improve the existing iterative searches for this specific problem. Finally, the usual way of implementing approximations of strict taboo search is discussed and better approximations are proposed.

Key words : Quadratic assignment problem, taboo search, genetic algorithms.

1.INTRODUCTION

In location science, many practical problems (see among others Burkard (1984, 1991), Burkard and Offermann (1985), Eiselt and Laporte (1991), Elshafei (1977), Finke et al. (1987), Laporte and Mercure (1988), Nugent et al. (1968), Steinberg (1961)) may be formulated as quadratic assignment problems (QAP). As this problem is NP-hard (Sahni and Gonzalez, 1976) and can be optimally solved for very small instances only, many heuristic methods have been developed. The problem may be stated as follows : n units have to be assigned to n different locations ; knowing that a flow of value a_{ij} has to go from unit i to unit j and that the distance between the locations r and s is b_{rs}, we want to find an assignment of the units to the locations that minimizes the sum of the products flow $\times$ distance. Mathematically, the problem can be stated as :

$$\min_{\pi \in P(n)} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi_i \pi_j}$$

where $\boldsymbol{A} = (a_{ij})$ and $\boldsymbol{B} = (b_{rs})$ are two $n \times n$ matrices, $P(n)$ is the set of all permutations of $\{1, \ldots, n\}$ and π_i gives the location of unit i in permutation $\pi \in P(n)$.

* Centre de recherche sur les transports
Université de Montréal
C. P. 6128, succursale Centre-Ville
Montréal, Canada H3C 3J7
e-mail : taillard@crt.umontreal.ca

This paper compares some of the most efficient heuristic methods. We focus here on methods that are able to find pseudo-optimal solutions of various practical problems in a reasonable amount of computation time. We mean by " pseudo-optimal solutions " solutions that we conjecture to be optimal, but for which the optimality is not proven. Indeed, exact procedures require computation times that are much higher than those needed by efficient heuristic methods (we are speaking of factors as large as 10^4 to 10^5 for a problem of size 20, see, e. g. Clausen and Perregård, 1994). Therefore, it is almost certain that a best known solution, repeatedly found (more than 30 times) with efficient heuristic methods, is an optimal one ; as the optimality is not proven, we introduce the term of pseudo-optimal solutions. The heuristic methods studied in this paper are not formally compared in the literature and since, in general, their performances strongly depend on the type of problem considered, the reader cannot easily determine which method is more appropriate for his application.

The methods we have chosen to consider in this paper are known as *Reactive tabu search* (Re-TS), due to Battiti and Tecchiolli (1994), *Genetic Hybrids* (GH), due to Fleurent and Ferland (1993), and *Robust taboo search* (Ro-TS), due to Taillard (1991), and are sketched in Section 2. We also consider approximations of the most elementary taboo search, corresponding to the very basic idea of taboo search principles as proposed by Glover (1989). We call *Strict taboo search* (S-TS) a method that simply forbids the repetition of already visited configurations.

In order to explain the differences in the efficiency of these methods when applied to various problems, we present and analyze the particularities of some of the problems commonly used in the literature in Section 3. Then, in Section 4 we compare the methods presented in Section 2 when applied to the problems of Section 3. In Section 5, we make a synthesis of the remarks made in the previous sections and we propose some improvements of the methods ; in particular, without restricting ourselves to the QAP, we show how to improve a usual way of implementing approximations of S-TS. Finally, on a new application of QAP to image synthesis, we illustrate how it is possible to adapt and speed up by a huge factor (i. e. several hundreds) the general methods described in this paper.

2.EFFICIENT ITERATIVE SEARCHES FOR THE QAP

We are not going to present all the existing heuristic methods for the QAP but only a selection of the most efficient ones. All of these methods are governed by taboo search principles and are presented in chronological order. Let us quote that the authors of these methods have reported all the best known solutions of classical problems (Skorin-Kapov (1990 and 1994), Taillard (1991)). Among other iterative searches we do not consider in

this section, let us quote the works of Chakrapani and Skorin-Kapov (1993), Voß (1993), Skorin-Kapov (1994) and Kelly et al. (1994).

2.1. Robust taboo search (Ro-TS), Taillard (1991)

Robust taboo search is certainly not the best one, but it is the simplest to implement among the methods presented in this section, since a procedure of less than two pages is sufficient to programme it. We briefly recall the basic principles of this local search method.

Let π be a current solution and $N(\pi)$ the set of all the permutations that can be obtained by exchanging two different elements of π : $N(\pi) = \{\mu \mid \mu_k = \pi_k \ \forall\, k \notin \{r, s\},\ \mu_r = \pi_s,\ \mu_s = \pi_r,\ 1 \le r < s \le n\}$. $N(\pi)$ is called the set of neighbour solutions of π. Let $\Delta(\pi, r, s)$ be the cost of exchanging units r and s located at π_r and π_s. It can be shown that :

$$\begin{aligned}\Delta(\pi, r, s) \;=\; & a_{rr}\,(b_{\pi_s\pi_s} - b_{\pi_r\pi_r}) + a_{rs}\,(b_{\pi_s\pi_r} - b_{\pi_r\pi_s}) + \\ & a_{sr}\,(b_{\pi_r\pi_s} - b_{\pi_s\pi_r}) + a_{ss}\,(b_{\pi_r\pi_r} - b_{\pi_s\pi_s}) + \\ & \sum_{k\,=\,1,\ k\,\neq\, r,\ s}^{n} \big(a_{kr}\,(b_{\pi_k\pi_s} - b_{\pi_k\pi_r}) + a_{ks}\,(b_{\pi_k\pi_r} - b_{\pi_k\pi_s}) + \\ & \quad a_{rk}\,(b_{\pi_s\pi_k} - b_{\pi_r\pi_k}) + a_{sk}\,(b_{\pi_r\pi_k} - b_{\pi_s\pi_k})\big)\end{aligned} \qquad (1)$$

Moreover, if μ is the solution obtained by exchanging units r and s in the solution π, it is possible to compute the value $\Delta(\mu, u, v)$ faster by using the fact that, for $\{u, v\} \cap \{r, s\} = \varnothing$ we have :

$$\begin{aligned}\Delta(\mu, u, v) \;=\; \Delta(\pi, u, v) + & (a_{ru} - a_{rv} + a_{sv} - a_{su})\,(b_{\mu_s\mu_u} - b_{\mu_s\mu_v} + b_{\mu_r\mu_v} - b_{\mu_r\mu_u}) + \\ & (a_{ur} - a_{vr} + a_{vs} - a_{us})\,(b_{\mu_u\mu_s} - b_{\mu_v\mu_s} + b_{\mu_v\mu_r} - b_{\mu_u\mu_r})\end{aligned} \qquad (2)$$

Very generally, a local search can be formulated as follows :

a) Choose an initial solution $\pi^{\mathbf{0}}$; set $k := 0$;

b) While a stopping criterion is not satisfied, repeat :

c) Choose a solution $\pi^{\boldsymbol{k+1}} \in N(\pi^{\boldsymbol{k}})$

$k := k + 1$

For the QAP, with the formulæ (1) and (2), it is possible to compute the values of all the solutions contained in $N(\pi^{\boldsymbol{k}})$ ($k \neq 1$) in $O(n^2)$ time. In Ro-TS, a random permutation is chosen as initial solution $\pi^{\mathbf{0}}$. At iteration k, the choice of $\pi^{\boldsymbol{k+1}}$, the next visited solution, is the best solution of $N(\pi^{\boldsymbol{k}})$ that is allowed, even if $\pi^{\boldsymbol{k+1}}$ is worse than $\pi^{\boldsymbol{k}}$, the current solution at iteration k. To be allowed, a solution must satisfy the following conditions (where t and u are two parameters) :

- If $k > t$ and $\pi_r^v \neq i$, $\forall v$, $k - t \leq v \leq k$ (i. e. if the number k of iterations performed is greater than t and the element i has never been at location r during the last t iterations), then the permutations of $N(\pi^k)$ that do not place i at location r are forbidden.
- If $\exists v, v \geq k - u$ such that $\pi_r^v = i$, $\pi_s^v = j$ and if $\pi_r^k \neq i$, $\pi_s^k \neq j$ (i. e. during the last u iterations, a solution had unit i placed at location r and unit j placed at location s), then it is forbidden to place both i at location r and j at location s again (unless this modification improves the quality of the best solution found so far).

The aim of parameters t and u is to prevent the algorithm from always visiting the same solutions. In Taillard (1991), it is shown that modifying u randomly during the search by choosing its value 10% around n, leads to a method that is fairly robust. The parameter t must be larger than $|N(\pi)|$; practically, values between $2n^2$ and $5n^2$ are perfectly convenient. It is clear that these values for t and u may be bad for some problems, but, in general, they produce acceptable results.

The effect of t is to diversify the search by imposing given solutions, even if their values are very high. So, this mechanism can be viewed as a long term memory. Conversely, the mechanism associated with u can be viewed as a short term memory.

2.2. The reactive tabu search (Re-TS), Battiti and Tecchiolli (1994)

As the reactive tabu search is also based on taboo search techniques, its working principle is the same. The initial solution is also chosen randomly, however, the diversification mechanism is different from the one used in Ro-TS : if given solutions are often visited, then several units are moved randomly and the other data structures are cleared.

The short term memory is also implemented with the forbidding of solutions, but the choice of the parameter u is different : if the search returns to a solution already visited, then the value of u is increased ; conversely, if the value of u is not changed during many iterations, then u is decreased. Naturally, many parameters regulate the mechanisms of reaction and diversification ; Battiti and Tecchiolli give standard values for these parameters, but we think that other values might be better for some types of problems.

For our numerical experiments, we have re-implemented a version of Re-TS ; this version is slightly different from the one of Battiti and Tecchiolli. It was not possible to use the code programmed by these authors because it was unable to solve asymmetrical problems and was coded in a programming language different from the one we have used to code the other methods. However, comparisons between our code and the one of Battiti and Tecchiolli have shown that the quality of the solutions produced by both codes are more or less similar.

2.3. Strict taboo search (S-TS), Battiti and Tecchiolli (1994)

Strict taboo search corresponds to the most basic taboo search where all the solutions of $N(\pi^k)$ that have been visited up to iteration k are forbidden. This method is easy to describe but hard to implement since all the visited solutions have to be recorded and all candidate solutions have to be compared to the recorded ones. This can be done efficiently using hashing techniques but the memory needed to store the visited solutions grows linearly with the number of iterations performed.

Battiti and Tecchiolli (1994) have implemented a S-TS in an efficient way (in $O(n^2)$ time per iteration) and have shown that S-TS requires less iterations than Re-TS to find pseudo-optimal solutions of random problems but the computation time may be higher. In the following, we will consider an approximation of S-TS and not a true S-TS : instead of storing each solution, an approximation of S-TS consists in computing a hashing function $h : P(n) \rightarrow \mathrm{Z}$ for each solution visited and to consider that a solution $\pi \in N(\pi^k)$ has been visited if $\exists i,\ k - u \leq i \leq k,\ h(\pi) = h(\pi^i)$, where u is a parameter. As many different solutions may have the same hashing value, this method is an approximation of S-TS.

In order to be able to test in $O(1)$ time whether a value has been computed or not, (if this test cannot be done in $O(1)$ time the complexity of one step of TS is increased), a vector $\boldsymbol{v}$ must be stored. The component k of this vector indicates whether the value k of the hashing function has already been computed or not. In practice, the length of $\boldsymbol{v}$ must be limited to a given length L (typically $10^4 \leq L \leq 10^6$). Then, the computation of the function h must be done modulo L. Several functions may be defined : Woodruff and Zemel (1993) propose $h_1(\pi) = \boldsymbol{z} \cdot \pi \bmod L$ (where $\boldsymbol{z}$ is a vector of pseudo-random integers). In the remainder, we test other hashing functions : $h_2(\pi) = \sum_i i^3 \pi_i \bmod L$ and $h_3(\pi) =$ (objective function value of π) mod L. This last hashing function was also used by Battiti and Tecchiolli in their implementation of Re-TS (with L set to infinity ; for our implementation of Re-TS, we have used $L = 200\,000$).

2.4. Genetic Hybrids (GH), Fleurent and Ferland (1994)

The basic ideas of genetic algorithms are the following (for more details see, among others, Davis (1987) and Holland (1976)) : at each step of the search, we have a *population* of solutions, that is to say a set of feasible solutions of the problem. One step consists in *selecting* two solutions of the population, the *parents*, and to mix these solution with a *crossover* operator in order to create a new solution, the *child*. Then, the child is randomly modified by a *mutation* operator and inserted in the population. Finally, the size of the new population is possibly decreased by a *culling* operator. There exist standard ways of

implementing selecting and culling operators. The crossover and mutation operators are the most difficult to define. For the QAP, the crossover operator used by Fleurent and Ferland (1994) is the one proposed by Tate and Smith (1995) in a " pure " genetic algorithm, and may be sketched as follows :

a) If an object is assigned to the same location for both parents, it remains at the same location for the child ;

b) Unassigned sites are scanned from left to right. For an unassigned site, an object is picked at random among those that occupied that site in the parent. Once an object is assigned, it is no longer considered in future random choices.

c) The remaining objects are assigned at random to the unassigned sites.

The mutation operator used in GH consists in applying few steps of Ro-TS to the child and to return to the best solution found during this search. There are several parameters in GH that need to be set and for some of them the way to do this is left open by the authors. For the numerical experiments of Section 4, we have chosen two parameter settings. For problems of small size (up to $n = 50$, first variant of the algorithm), we limit the size of the population to $2n$ and for larger problems (second variant) we limit the size of the population to 100 and the total number of Ro-TS iterations to $1000n$. The reasons for limiting the population size to 100 are first that the authors of GH propose this population size and second that having a larger population (or larger than $2n$ for problems with $n \leq 50$) does not improve the performance of the algorithm. More precisely, the parameter settings we have chosen are the following :

- The initial population is obtained by choosing random permutations that are passed through the mutation operator.
- The size of the population is $2n$ (or 100 for the second variant).
- The ith worst solution of the population has a probability $2i/p(p + 1)$ of being selected (where p is the current size of the population).
- The mutation operator performs $4n$ iterations of Ro-TS.
- the culling operator eliminates the two worst solutions of the population every two steps.
- The stopping criterion is met when all the solutions of the current population have the same value or when a pseudo-optimal solution is found (or when the procedure has performed a total number of $1000n$ iterations of Ro-TS for the second variant).

Note that, for the first variant, it is possible that the process takes long to stop. Probabilistically, it can be shown that the process always stops. In practice, we have never observed extremely long runs (see e. g. Tables 2 and 3).

3.PROBLEMS

We will see that the efficiency of the methods presented above very strongly depends on the type of problem being solved. Hence, an excellent method for a given problem type may be inefficient for another type. Moreover, very few authors compare their method with other ones on many different problem types. This is what has motivated us to present the characteristics of a representative sample of problems coming from the literature before comparing the various methods on this sample. Most of the problems presented in this section may be found in the QAPLIB library of Burkard et al. (1991).

3.1. Random and uniform distances and flows.

The problems considered in this paragraph are randomly generated from a uniform distribution. Very often, the bounds between which the distances and flows are generated are 0 and 99. This type of problem has been widely used : Roucairol (1987) has proposed 4 small problems of size n = 10, 12, 15 and 20 ; Taillard (1991) has proposed a set of 18 problems of size 5 to 100 that have been used by Battiti and Tecchiolli (1994) and Fleurent and Ferland (1994) to measure the performances of their algorithms. From now on, we will denote these problems by *Tai*nn*a*, where nn is the size of the problem considered.

Among the problems considered in this section, this type of problem is certainly the most difficult to solve optimally. Indeed, in Taillard (1991) we succeeded in solving pseudo-optimally the problems of size less or equal to 35, but it would be very hazardous to consider that the optimum of the problems of larger size have been found.

Though these problems are very difficult to solve (pseudo-) optimally (see e. g. Taillard, 1991), they are generally well handled by iterative searches, in the sense that all these methods find solutions 1 or 2 per cent above the pseudo-optimum in a short computation time. This paradox — it is difficult to find the optimum but simple to find a good solution — may be explained by the fact that the different local optima are very slightly correlated (see the entropy measure in § 3.4) but that their evaluation is contained between tight bounds ; this is the direct consequence of a result obtained by Burkard and Fincke (1985) who have shown that, with a probability tending to 1, the relative difference between the best and the worst solution of this type of problems tends to 0 when the problem size tends to infinity.

These problems have been widely used in the literature because they are almost the only ones that have not been solved pseudo-optimally. However, even if they are not optimally solved and if it is still possible to glean minute improvements, we think that these problems are not interesting : all the recent heuristics find good solutions and the fact of

having improved one of the best known solutions does not constitute a proof of the efficiency of a method.

3.2. Random flows on grids

The problems considered in this paragraph are generated as follows : a rectangular tiling constituted of $n_1 \times n_2$ squares of unit size is considered. A location is one of these squares and the distance between two squares is the Manhattan distance between them (i. e. the distance between two squares located at (i, j) and (r, s) (Cartesian coordinates) is $|i - r| + |j - s|$). The flows are randomly generated, but not necessarily uniformly.

These problems have the particularity of being symmetrical and hence to have multiple of 4 ($n_1 \neq n_2$) or 8 ($n_1 = n_2$) different optimal solutions. The problems of Nugent, Vollmann and Ruml (1968) (Nug*nn*), Skorin-Kapov (1990) (Sko*nn*) and Wilhelm and Ward (1987) are of this type. Pseudo-optimal solutions of this family of problem may be found up to a size of 64.

3.3. Real-life problems

We have grouped in this paragraph isolate problems arising from practical applications ; we are not going to make a complete list of real-life application of the quadratic assignment problem, but we only present the instances of problems that are often used in the literature. Moreover, we introduce a new type of problem occurring in the field of image synthesis.

a) Steinberg's problem (1961)

There are three versions of this problem whose goal is to minimize the length of connections between units that have to be placed on a rectangular grid : the first version has a distance matrix corresponding to Manhattan distances, the second the square of Euclidean distances and the third Euclidean distances. The flow matrix (thrice the same) gives the number of connections to make between the units. The size of the problem is $n = 9 \times 4 = 36$. They are denoted *Ste*36*a*, *Ste*36*b*, *Ste*36*c* and they are solved pseudo-optimally.

b) Elshafei's problem (1977)

This problem has been created to find the sites on which the various units of a hospital have to be placed. The placement of the units has to minimize the total distance daily performed by the users of the hospital. The distance matrix corresponds to the Euclidean distances between the sites (with a penalty when the sites are not located on the same floor) and the flow matrix corresponds to the displacement intensity of the users. The size of the problem is $n = 19$. It is denoted *Els*19 and it is solved optimally.

c) Burkard and Offermann's problems (1977)

The goal of this problem type is to find out what would be, in theory, the best typewriter keyboards for various languages and for mechanical or electrical machines. The distance matrix corresponds to the time between the typing of two keys (the time depends on the fact that the machine is an electrical or a mechanical one) and the flow matrix contains the frequencies of appearance of two letters in a given language. As four different languages and two typewriters are considered, there are 8 problems of this type. The size of the problems is $n = 26$. They are denoted *Bur*26*a*, …, *Bur*26*h* and they are solved pseudo-optimally.

d) Density of grey

In order to print a grey of density of m/n, a technique consists of generating a rectangular grid containing $n = n_1 \times n_2$ square cases with m black cases and $n - m$ white cases. By juxtaposing many of these grids, one gets a grey surface of density m/n. To get the finest frame, the black (or white) cases have to be spread as regularly as possible on the grid. To present this problem as a QAP, we can consider m electrons that have to be put on the cases. The placement must be done in such a way that the sum of the intensities of the electrical repulsion forces is minimized.

In the problems we are going to treat, we have considered the forces f_{rstu} ($r, t = 1, \ldots, n_1, s, u = 1, \ldots, n_2$) between the two cases i and j located at the coordinates (r, s) and (t, u) :

$$f_{rstu} = \max_{v,\, w \in \{-1,\, 0,\, 1\}} \frac{1}{(r - t + n_1 v)^2 + (s - u + n_2 w)^2}$$

To get a grey density of m/n, the QAP with the following matrices can be solved :

$$a_{ij} = \begin{cases} 1 \text{ if } i \le m \text{ and } j \le m \\ 0 \text{ otherwise} \end{cases}, \quad b_{ij} = b_{n_2 (r-1) + s \quad n_2 (t-1) + u} = f_{rstu}.$$

The ith component ($i \le m$) of a solution π, $\pi_i = \pi_{n_2 (r-1) + s}$ gives the location where a black case has to be placed in the grid.

The problems of this type are relatively simple, but some methods may be trapped by the fact that many solutions with the same objective value exist : exchanging two black (or two white) cases or performing a symmetry, rotation or shifting of the cases does not change the value of the solutions. These problems are denoted *grey*n_1_n_2_m and are solved pseudo-optimally for $n_1 = n_2 = 8$. They constitute a new application of QAP that has not yet been presented to our knowledge. By choosing other definitions of distances and by varying the choices of n_1, n_2 and m it is possible to create many different problems.

3.4. Characteristics of real-life problems

Real-life problems are very different from randomly generated ones as those presented in §3.1 and §3.2. The first remark that can be made is that the flows matrices have a large number of zero components and the remaining ones are clearly not uniformly distributed. In Figure 1, we give, for the problems of Steinberg, Elshafei and Burkard and Offermann (for German language) the distribution of the values (normalized between 0 and 1) encountered in the flow matrices.

The second remark that can be done pertains to the structure of local optima. We say that a solution is a local optimum if it is not possible to improve the quality of this solution by exchanging the location of two units only. It turns out that real-life problems have some structure that can be found by examining local optima. In other words, the permutations corresponding to local optima privilege given locations for given units. This may be measured by the *entropy* of a set of permutations, as defined by Fleurent and Ferland (1994) : Let m be the number of solutions considered and n_{ij} the number of times that the unit i is located on site j in the m solutions. Let us define the values

$$v_{ij} = \begin{cases} 0 \text{ if } n_{ij} = 0 \\ \frac{-n_{ij}}{m} \log \frac{n_{ij}}{m} \text{ otherwhise} \end{cases}$$

The entropy E of the m solutions is given by :

$$E = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij}}{n \log n}$$

So, a set of identical solutions (permutations) has an entropy of $E = 0$ and a set of solutions uniformly distributed in the set of permutations of n elements has an entropy tending to 1 when m grows to infinity.

Empirically, to show that real-life problems have more structure than uniformly generated ones, we consider the local optima that are obtained from 10 000 random solutions to the Elshafei's problem. The initial solutions have an entropy $E = 0.9996$ and the local optima have an entropy $E = 0.8$. For uniformly generated problems like those of §3.1, the entropy of local optima is $E = 0.97$ for $n = 19$ and $E = 0.996$ for $n = 50$.

To illustrate this more intuitively, we give, in Figure 2, as a function of k, the proportion of solutions having k units placed at the same location. We see that this proportion decreases very fast when the solutions are drawn from a uniformly distributed population ; the decrease is slower when the permutations are drawn from local optima of random problems and that a relatively high proportion of local optima of the Elshafei's

problem differ by few units. Let us quote that 1/50 of the local optima of Elshafei's problem are global optima ; when starting with initial solutions such that 3 given units are placed where they are located in a global optimum and placing randomly the other ones, we have observed that about one half of the local optima were global optima. This means that, when some information on the locations of few units is known, this problem becomes very easy.

3.5. Non-uniform, random problems

The size of real-life problems is relatively small (except for the problems presented in § 3.3d) ; it is therefore difficult to compare the methods on an " interesting " type of problems over various sizes — we have seen that the random problems used in the literature are not really interesting. So, we have tried to construct problems having about the same characteristics as those of real-life problems. These problems are randomly generated, but not uniformly : the distance matrix corresponds to the Euclidean distance between n points of the plane. The locations of these points are generated by calling as many times as needed the following procedure :

- Choose Θ randomly, uniformly between 0 and 2π.
- Choose R randomly, uniformly between 0 and M.
- Choose N randomly, uniformly between 1 and K
- Repeat N times :

  Choose θ randomly, uniformly between 0 and 2π.

  Choose r randomly, uniformly between 0 and m.

  The Euclidean coordinates of the next generated point are :
  $(R \cos \Theta + r \cos \theta, \quad R \sin \Theta + r \sin \theta)$

Where M, K and m are parameters that allows to generate various problems types : (0, 1, m) generates the points uniformly in a circle of radius m ; (1000, 20, 10) generates clusters of 1 to 20 points that are uniformly distributed in a circle of radius 1000, the points in the clusters being distributed in a circle of radius 10.

The matrix of flows is also randomly generated but not uniformly. Let X be a random variable uniformly distributed between 0 and 1 ; the flow a_{ij} between units i and j is given by : $a_{ij} = \lfloor 10^{((B-A)X + A)} \rfloor$ where A and B are parameters such that $A < B$ and $B > 0$; these parameters allow to generate various densities of flows : $A = -10$ and $B = 5$ generates a matrix containing about 2/3 of zero entries and flows having a maximal value of 10^5, that is to say something near to what is observed for the Elshafei's problem. These problems are denoted *Tai*n*n**b* and the instances we have generated are solved pseudo-optimally for $n \leq 50$.

4.COMPARISON OF THE METHODS

4.1. Direct comparison based on published results

The first analysis that can be done is to look at the numerical experiments done by the authors of the methods. First, Battiti and Tecchiolli (1994) show that S-TS and Re-TS find the pseudo-optimal solutions of Tai*nn*a problems in less iterations than Ro-TS (when $n \leq 25$). Fleurent and Ferland (1994) show that GH finds better solutions than Ro-TS on Sko*nn* problems ; more precisely, for problem Sko100a, repeating 1000 times short Ro-TS ($4n$ iterations) is worse than performing one long Ro-TS ($4000n$ iterations) which itself is worse than a GH that calls 1000 times a mutation operator that performs $4n$ iterations of Ro-TS. Considering the best solutions found, let us mention that the authors of GH have slightly improved one solution (Sko81) of the set Sko42, …, Sko90, regarding to the values found by Ro-TS.

Finally, Ro-TS, Re-TS and GH may be directly compared on the base of the best solutions they have found on problems Tai40a, …, Tai100a ; in Table 1, we give the values published by the authors of these methods, as well as the best known solution value of these problems. The best known solutions of problems Tai40a and Tai100a has been communicated to us by Roberto Battiti (october 13. 1994) and we have found the best known solutions of Tai60a and Tai80a during the computational experiments performed for the present paper.

Problem	Ro-TS	Re-TS	GH	Best known
Tai40a	3146541	3141702	3141702	3139370
Tai50a	4951186	4948508	4941410	4941410
Tai60a	7272020	7228214	7254564	7208572
Tai80a	13582038	13558710	13574084	13557864
Tai100a	21245778	21160946	21235482	21125314

Table 1 : Best solutions founds by the authors of Ro-TS, Re-TS and GH on Tai*nn*a problems.

For this type of problems, Re-TS is the best method, even if GH has found a solution better than the other methods for Tai50a. Looking at this table, it could be thought that GH is better than Ro-TS, but as we will see (tables 2 and 4), this is not true (at least for " short " searches that perform less than $1000n$ iterations). As the authors of Re-TS give results on QAP for Tai*nn*a problems only, it is not possible to directly compare the methods on other problems.

4.2. Comparison of approximations of S-TS and Ro-TS

In this paragraph, we compare Ro-TS and the approximations of S-TS with hashing functions $h2$ and $h3$ (c. f. §2.3). Here, we only consider short term comparisons ; longer term comparisons are done in § 4.3. To make these short term comparisons, we proceed as follows : the initial solutions given to the various methods are the best known solutions to which m random exchanges of two units are performed. After $2n$ iterations, if a process has not found the best known solution again, we consider that it is lost. In Figures 3 to 5, we give, as a function of m, the supplementary (regarding to Ro-TS, and possibly negative) proportion of searches that have found the best known solution. We give these proportions for the approximations of S-TS with the hashing functions $h2$ and $h3$ and for a simple local search that stops at the first local optimum found (steepest descent).

Figure 3 gives these proportions for the problem Tai50a ; we see that both approximations of S-TS are much better than Ro-TS since, for S-TS, the number of searches refinding the best known solution may be almost 40% higher than for Ro-TS. There is almost no difference between the approximations with $h2$ or $h3$; the descent method is much less efficient than Ro-TS.

Figure 4 gives these proportions for the problem Nug20 ; the situation is completely different : the approximation of S-TS with $h2$ produces much better results than the approximation with $h3$ which is not clearly better than Ro-TS (the descent method is still worse than Ro-TS). This can be explained by the fact that the optimum of this problem has a value of 2570 ; consequently, $h3$ can not take a large number of different values. Therefore many unvisited solutions are forbidden in the approximation of S-TS with $h3$.

Figure 5 gives these proportions averaged for the problems Bur26a, …, Bur26h. For these problems, the approximation of S-TS with $h3$ produces slightly better results than Ro-TS and the approximation with $h2$ is worse than Ro-TS (once again, the descent method is the worst). This can be explained by the fact that these problems have many local optima with the same value. Consequently, when the search falls into a local optimum, $h3$ prevents it from visiting all the other local optima with the same value while $h2$ allows it to visit a large number of different local optima with the same value.

4.3. Comparison of Ro-TS, approximations of S-TS, Re-TS and GH on various problems

In order to compare these methods, we have programmed Re-TS and GH. Therefore, the computational results given in this paragraph also constitute a verification of the performances announced by their authors. The first version of GH we consider is the one

with a population of size 2n that stops when all the solutions of the population have the same value or when a pseudo-optimal solution is found.

For the approximations of S-TS, we have considered a vector of length L = 200 000 that records the values computed by the hashing functions and we forbid during 10 000 iterations to visit a solution whose hashing value has already been computed. All the taboo searches are stopped when they have performed the same number of iterations as the average number of iterations performed by the Ro-TS procedure included in GH. The speed difference between all the taboo searches is negligible. As the time spent by GH out

Problem	Number found (GH)	Number of iterations	Nr. of iter., optimum not found	Ro-TS	S-TS ($h2$)	S-TS ($h3$)	Re-TS	GH
Nug20	50/50	2'512	—	0.037	0.037	2.604	0.003	0
Nug30	18/30	21'872	35'160	0.026	0.148	2.182	0.030	0.026
Sko42	28/30	45'271	78'120	0.013	0.180	1.719	0.018	0.005
Tai20a	8/30	13'043	16'124	0.235	0.265	0.343	0.312	0.350
Tai25a	4/30	28'913	30'562	0.304	0.222	0.245	0.247	0.428
Tai30a	9/30	46'864	51'691	0.326	0.187	0.184	0.186	0.265
Tai35a	5/30	82'264	85'814	0.546	0.259	0.227	0.310	0.431

Table 2 : Comparison of the methods on random problems.

of the Ro-TS procedure is also negligible, the computation times for all methods may be considered as identical.

In Tables 2 and 3, we compare Ro-TS, Re- TS, the approximations of S-TS with $h2$ and $h3$ and GH. In these tables, we give :

a) the problem considered,

b) the proportion of GH searches that have found the pseudo-optimum,

c) the number of iterations performed by the taboo searches,

d) the mean number of Ro-TS iterations performed in the mutation operator of GH when this algorithm does not find the pseudo-optimal solution,

e) the relative performances of these methods measured in per cent above the pseudo-optimum value.

In Table 2, we compare these methods when applied to the randomly generated problems Nugnn, Sko42 and Tainna. In this table, we see that the GH method produces very good solutions for the problems Nugnn and Skonn ; moreover, the number of successful searches is very high. The approximations of S-TS are the worst methods for this type of problems ; the approximation with $h3$ is very bad because of the low values of the solutions. For uniform random problems, we see that the conclusions are reversed : the best

methods are the approximations of S-TS, followed by Re-TS ; Ro-TS and GH are the worst methods for these problems (notice that the performances of Ro-TS can be notably improved by increasing the value of parameter t from $3n^2$ (used in Tables 2, 3 and 4) to a higher value).

Without taking into account the approximation of S-TS with $h3$, we can say that all the problems of this table are very well solved, since all the solutions found are, on average, less than 1% above the best known solution. It is interesting to observe that these results are very similar to those observed in Figures 3 and 4. So, the method described in § 4.2 for fast tuning of the short term memory parameters seems to be valid.

In Table 3, we give the same statistics for real-life problems and we see that the situation is totally different : for this type of problems GH is the best method since it finds pseudo-optimal solutions in most cases and the average solutions found are less than 0.05% above the best known solutions — so, real-life problems seem to be easier than those of Table 2 — ; the second best method is Ro-TS (with two exceptions), followed by Re-TS ; finally, the approximations of S-TS are clearly not adapted for this type of problem (the approximation with $h3$ for grey problems excepted, since this approximation prevents from performing useless modifications of the solutions). In this table, when many problems of the same type and size exist, we give average values.

Problem	Number found (GH)	Number of iterations	Nr. of iter., optimum not found	Ro-TS	S-TS ($h2$)	S-TS ($h3$)	Re-TS	GH
Bur26a-h	737/800	7734	23028	0.035	0.370	0.260	0.560	0.004
Ste36a-c	76/90	36626	76298	0.053	4.908	6.205	0.817	0.023
Grey8_8_13	30/30	3892	—	0.439	0.703	0	0.335	0
Els19	99/100	2166	7144	9.177	24.739	24.436	7.836	0.042
Tai20b	29/30	3675	9280	0.540	16.770	16.947	3.919	0.015
Tai25b	30/30	9473	—	0.011	17.008	17.509	1.021	0
Tai30b	30/30	22048	—	0.307	12.514	12.512	1.101	0
Tai40b	30/30	31595	—	0.744	10.268	10.377	1.330	0
Tai50b	20/30	124867	163080	0.228	6.213	6.221	0.409	0.026

Table 3 : Comparison of the methods on real-life problems and real-life like problems.

For real-life problems, we see that the diversification of the search is very important because many poor local optima exist and the approximations of S-TS, that do not have such a mechanism, are frequently trapped in these local optima. For the smallest problems (Els19, Tai20b), the diversification mechanisms of Ro-TS and Re-TS did not have time

enough to fully produce their effects. To our knowledge, the best method for solving the problem Els19 is to repeat descent methods : with this algorithm an optimal solution is found in about 1000 iterations on average (while GH needs more than two times this number of iterations).

Our explanation of the performances of GH is that this method succeeds in finding out and exploiting the structure (if there is any) of the problems. As purely random problems have no structure, it works poorly on these problems, conversely, as real-life problems are more structured, it works very well on these problems ; more precisely, GH works well on problems for which the local optima entropy is low. If GH seems to be the most reliable method, it cannot be recommended for all cases since its running time is very high (this is the reason why we do not treat larger problems in Tables 2 and 3).

Problem	Ro-TS	S-TS ($h2$)	S-TS ($h3$)	Re-TS	GH	Best known value
Tai50a	1.104	0.985	0.795	0.952	1.352	4941410
Tai60a	1.278	0.918	0.696	0.859	1.340	7208572
Tai80a	0.961	0.718	0.432	0.569	1.106	13557864
Tai100a	0.823	0.690	0.295	0.387	1.139	21125314
Sko49	0.096	0.063	1.394	0.068	0.120	23386
Sko56	0.090	0.408	1.614	0.145	0.181	34458
Sko64	0.063	1.548	1.548	0.125	0.174	48498
Sko72	0.181	0.248	1.459	0.110	0.200	66256
Sko81	0.088	0.247	1.446	0.110	0.250	90998
Sko90	0.179	0.216	1.552	0.164	0.314	115534
Sko100a-f	0.162	0.323	1.406	0.141	0.264	150252.7
Tai50b	0.439	7.567	7.689	0.731	0.224	458821517
Tai60b	0.899	9.297	9.298	0.366	0.115	608215054
Tai80b	1.004	5.192	5.205	1.800	0.597	818415043
Tai100b	0.968	6.042	6.045	1.490	0.241	1185996137
Tai150b	1.904	2.903	2.909	0.807	0.941	499348972

Table 4 : Comparison of the methods on larger problems.

In Table 4, we compare the same methods on larger problems, but with another version of GH : the size of the population is fixed to 100 and the number of calls of the mutation operator (performing $4n$ steps of Ro-TS) is limited to 250. For the taboo searches, the number of iterations is limited to $1000n$. Again, the computation times of the various methods can be considered as equivalent. In this table, we give the results in per cent above the best known solution. Again, the results are averaged when several problems of the same size and type are solved.

We see that the best method for uniform random problems (Tai*nn*a) is the approximation of S-TS with *h*3 (while this method is the worst for the remaining problems) and the worst is GH ; for the problems of Skorin-Kapov (Sko*nn*), the best method seems to be Re-TS, followed by Ro-TS (in Table 3, we have seen that GH was the best method for this type of problems ; this means that GH needs long computation times to be competitive) ; for real-life like problems (Tai*nn*b), the best method is GH and both approximations of S-TS are very bad since they produce solutions from 3 to 9 per cent above the best known solutions.

5.SYNTHESIS

In Section 4, we have seen that the efficiency of a method strongly depends on the type of problem treated. No single method is better than all the others for all the problems. Consequently, it is necessary to adapt the method, or to use the appropriate method for the specific problem type that is to be solved. The approximation of S-TS seems to be the best for uniform random problems ; GH seems to be very robust (especially for real-life problems) but is not competitive on uniform random problems ; moreover, GH requires a very long computation time. Therefore, if good solutions are needed in a short amount of time, we suggest to use Ro-TS, Re-TS or even multiple steepest descents that start from various initial solutions. In this section, we will discuss some slight modifications that might improve the methods studied in this paper. It is not our aim to discuss modifications that would radically change the methods, such as the use of other intensification or diversification mechanisms.

The first adaptation we have tried is to modify the GH algorithm in such a way that it produces better results on random problems. The first idea consists in changing the mutation operator that uses Ro-TS by another one, better adapted to this type of problems, for example an approximation of S-TS. Unfortunately, S-TS provides better results than Ro-TS after an extensive computational effort and the performance of GH may be only slightly improved by using a better mutation operator. We think that the simplest way of improving the methods studied is to tune their various parameters. As this is fastidious and does not present a great interest for this paper, we do not give any results in this regard.

First, we show how it is possible to speed-up and to notably improve all these methods for the problem of generating grey patterns by considering its specificities. We then present a way to improve the approximations of S-TS by using multiple hashing functions.

5.1. An efficient taboo search for grey densities

The problems of grey densities are well solved with the approximation of S-TS with $h3$ (cost of the solution) because this approximation does not allow the permutation of two black or two white cases. So, for this problem, the neighbourhood must be restricted to the exchange of one of the first m units (black) with one of the last $n - m$ units (white). Therefore, the neighbourhood size decreases from $O(n^2)$ time to $O(m(n - m))$ time. Moreover, the computation of the value of a solution in the neighbourhood can be done faster because the flow matrix contains 0 and 1 entries only. The simplified formulæ giving the value of exchanging two units (see equations (1) and (2), §2.1) become :

$$\Delta(\pi, r, s) = 2 \cdot \sum_{k=1}^{m} b_{\pi_k \pi_s} - b_{\pi_k \pi_r} \qquad (3)$$

and

$$\Delta(\mu, u, v) = \Delta(\pi, u, v) + 2 \cdot (b_{\mu_s \mu_u} - b_{\mu_s \mu_v} + b_{\mu_r \mu_v} - b_{\mu_r \mu_u}) \qquad (4)$$

For this type of problems, we have adapted Ro-TS, making these simplifications. The resulting method, for the problem Grey8_8_13, needs 13 times less iterations than the original GH algorithm to find the best known solution ; moreover, one iteration, for this problem, takes 23 times less computation time ; this means that the resulting speed-up has a value of more than 300. This allows to treat much larger problems. In Figure 6, we give four frames we have found to create greys of densities 11/256, 30/256, 43/256 and 98/256 as well as the frames usually used (see Ulichney (1987)) for the same densities. The reader can judge the difference of quality of both frames.

The computation of a complete chart of 256 greys takes some hours on a personal work-station. Nowadays, problems of this size cannot be treated by general methods for QAP.

5.2. Improving the approximations of S-TS

A very easy way of implementing an approximation of S-TS is to use an integer vector $\boldsymbol{v}$, $|\boldsymbol{v}| = L$, whose component i indicates the last iteration at which the value i of a hashing function has been computed. Doing like this, it is possible to efficiently test the taboo status of a solution (i. e. a solution with a hashing value i is forbidden if v_i is larger than the current iteration number minus u, a parameter). The solutions visited are not stored, so, the memory used does not depend on the number of iterations performed and the implementation of the method is very simple. Naturally, the search is not a true S-TS any more, but an approximation of S-TS.

A good hashing function should uniformly distribute the solutions over the interval $[0, L[$. A usual measure of the quality of a hashing function h is given by the probability

of collision, defined as $P\{h(\boldsymbol{x}) = h(\boldsymbol{y})\}$, where $\boldsymbol{x} \neq \boldsymbol{y}$. This probability must be as low as possible, $1/L$ ideally. However, we think that this measure is not appropriate in the case of the approximations of S-TS. Instead of this measure, we propose to consider the statistical expectation of the number of solutions that have to be generated before having a solution with a hashing value already computed. If one has a perfect hashing function, the probability $p(i)$ of generating a solution whose value is identical to one of the $i-1$ preceding solutions is (each of the $i-1$ preceding solutions is supposed to have a different hashing value) :

$$p(i) = \frac{i-1}{L} \cdot \left(1 - \sum_{k}^{i-1} p(k) \right), \; p(1) = 0$$

The new measure E^L we propose to introduce is given by : $E^L = \sum_{i=1}^{L} i \cdot p(i)$.

Figure 7 represents E^L as a function of L and the empirical expectation of the number of permutations that can be randomly generated before finding one having the same value of $h1$ or $h2$. For $h1$, the random numbers z_i were chosen between -10^6 and 10^6. We see that both functions are almost ideal when $n \geq 30$; for smaller n, it is better to choose $h1$.

E^L can be roughly approximated by $\frac{5}{4}\sqrt{L}$ (some theoretical values are : $E^{10^4} \cong 126$, $E^{10^6} \cong 1254$, $E^{10^8} \cong 12534$). So, even if the probability of collision is 10^{-6} for $L = 10^6$, (in practice this corresponds to a large vector), an approximation of S-TS applied to a problem of size $n = 100$ forbids about 5 solutions in mean at the first iteration while the exact S-TS forbids only one solution. Moreover, this proportion grows very fast with the number of iterations performed. Consequently, the choice of a unique hashing function, even an ideal one, leads to very rough approximations of S-TS.

The method we propose for improving the ratio of the expectation of the number of solutions generated before generating a solution with the same hashing value over the memory required is to use q different hashing functions $h^1, \ldots, h^q$ (therefore q vectors $\boldsymbol{v}^k$, $k = 1, \ldots, q$ of size L) ; in the approximation of S-TS, a solution π is considered to be visited if the q values of the vectors $v^1(h^1(\pi)), \ldots, v^q(h^q(\pi))$ indicate that $h^1(\pi), \ldots, h^q(\pi)$ have all been computed once already.

We give in Figure 8, as a function of q, the expected value E^{qL} of the number of solutions that have to be generated before finding q hashing values already computed ; we give these expectations for various total memory sizes qL. We see that these curves grow very fast with q, then they decrease slowly. In practice, $3 \leq q \leq 6$ provides good results, considering the memory used and the computation times of the hashing functions. With $n = 30$

and hashing functions of type h_2, we have observed results in accordance with the theoretical curves of Figure 8.

Empirically, the best efficiency of E^{q*L} ($q*$ being the best q for a total memory size qL) grows in $O((q*L)^{0.87})$ (the power has been evaluated by interpolation). This is to compare with the $O(L^{1/2})$ when q is set to 1. Consequently, this technique allows to render an approximation of S-TS more precise in the sense that collisions are less frequent.

6.CONCLUSIONS

To conclude, we wish to make some remarks that might help the reader to design a local search type method adapted for the special problem he wants to solve. Indeed, we have seen that none of the methods studied here is better than all the others for every type of problem. Even the most elementary TS may be better, for some specific problems, than more sophisticated methods.

First, the designer must observe the data of the problem in order to discover its particularities, he has to see whether the local search method is likely to be trapped in bad local optima. We have illustrated this with the (simple) example of designing grey patterns ; the particularities of this problem are easy to find but without taking them into account, it is not possible to create a good chart of 256 grey levels.

Second, if many approaches are possible for the short term direction of the search, we have seen that it is possible to almost surely determine the best one by examining the proportion of searches that succeed in refinding a good solution when this solution, slightly modified, is provided as initial solution to the search. If, for the problem considered, it is not efficient to perform multiple steepest descents that start with randomly generated solutions, then we have seen that a method like Ro-TS is extremely simple to implement (moreover the code of this procedure is public) and reasonably reliable.

Then, if the quality of the results provided by Ro-TS is not good enough, we suggest to implement a method like GH. Indeed, using a population of size $2n$ and a mutation operator that performs $4n$ iterations of a Ro-TS, the mean solutions produced by GH have a value less than 1% above the best known solution for all the problems considered in this paper. However, the reader must be aware that GH requires a great computational effort since its initialization takes a time proportional to n^4. Therefore, its complete execution may not be in accordance with a rapidity criterion.

Finally, let us say that an approximation of S-TS seems to be easy to implement and efficient for problems that are not really structured. For such problems, Re-TS is also an efficient method, but it is harder to implement. The approximation of S-TS can be improved by using many hashing functions at the same time.

In the future, we think that the designers of new local search methods should pay attention to the use of informations that may be gathered during the search. Indeed, one can consider that the methods studied in this paper waste informations they could use : Ro-TS does not exploit informations on the solutions visited and stores only one information for each move performed ; although S-TS stores all the solutions visited, this method does not exploit these informations : it even makes no use of their quality ; a similar remark might be made for Re-TS ; only GH gathers informations, via a population of solutions, but it stores only one solution every 4*n* iterations.

Acknowledgements

The author would like to thank the anonymous referees, Roberto Battiti and Gilbert Laporte whose valuable and multiple suggestions have improved the presentation of this paper. This work was partly supported by the National Science and Engineering Research Council of Canada, in the Canada International Post-Doctoral Fellowship Program.

REFERENCES

R. Battiti and G. Tecchiolli, (1994), *The Reactive Tabu Search*, ORSA Journal on Computing 6, 126–140.

R. E. Burkard, (1984), *Quadratic assignment problems*, European Journal of Operational Research 15, 283–289.

R. E. Burkard, (1991), " Location with spacial interactions : the quadratic assignment problem ", in P. B. Mirchandani and R. L. Francis (editors), *Discrete Location Theory*, John Willey, Berlin.

R. E. Burkard and U. Fincke, (1985), *Probabilistic asymptotic properties of some combinatorial optimization problems*, Discrete Applied Mathematics 12, 21−29.

R. E. Burkard, S. Karisch and F. Rendl (1991), *QAPLIB — A quadratic assignment problem library*, European Journal of Operational Research 55, 115–119.

R. E. Burkard and J. Offermann, (1977), *Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme*, Z. Operations Res. 21, B121−B132.

J. Chakrapani and J. Skorin-Kapov, (1993), *Massively parallel tabu search for the quadratic assignment problem*, Annals of Operations Research 41, 327–341.

J. Clausen and M. Perregård, *Solving Large Quadratic Assignment Problems in Parallel*, DIKU, Department of Computer Science, niversity of Copenhagen, 1994.

L. Davis (editor), (1987), *Genetic Algorithms and Simulated Annealing*, Pitman, London.

H. A. Eiselt and G. Laporte, (1991), *A Combinatorial Optimization Problem Arising in Dartboard Design*, Journal of the Operational Research Society 42, 113–118.

A. E. Elshafei, (1977), *Hospital layout as a quadratic assignment problem*, Operations Research Quarterly 28, 167−179.

G. Finke, R. E. Burkard and F. Rendl, (1987), *Quadratic assignment problems*, Annals of Discrete Mathematics 31, 61–82.

C. Fleurent and J. A. Ferland, (1994), *Genetic Hybrids for the Quadratic Assignment Problem*, DIMACS Series in Mathematics and Theorical Computer Science 16, 173–187.

F. Glover, (1989), *Tabu Search — Part I*, ORSA Journal on Computing 1, 190–206.

J. H. Holland, (1976), *Adaptations in Natural and Artificial Systems*, University of Michigan press, Ann Arbor.

J. P. Kelly, M. Laguna and F. Glover (1994), *A study of diversification strategies for the quadratic assignment problem*, Computers and Operations Research 21, 885–893.

G. Laporte and H. Mercure, (1988), *Balancing hydraulic turbine runners : A quadratic assignment problem*, European Journal of Operational Research 35, 378–381.

Ch. E. Nugent, Th. E. Vollmann and J. Ruml, (1968), *An experimental comparison of techniques for the assignment of facilities to locations*, Operations Research 16, 150−173.

C. Roucairol, (1987), *Du séqentiel au parallèle : la recherche arborescente et son application à la programmation quadratique en variable 0 et 1*, thèse d'État, Université P. & M. Curie, Paris.

S. Sahni and T. Gonzalez, (1976), *P-complete approximation problems*, Journal of the ACM 23, 555–565.

J. Skorin-Kapov, (1990), *Tabu search applied to the quadratic assignment problem*, ORSA Journal on Computing 2, 33−45.

J. Skorin-Kapov (1994), *Extension of a tabu search adaptation to the quadratic assignment problem*, Computers and Operations Research 21, 855–865.

L. Steinberg, (1961), *The backboard wiring problem : a placement algorithm*, SIAM Review 3, 37−50.

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

D. E. Tate and A. E. Smith, (1995), *A genetic approach to the quadratic assignment problem*, Computers and Operations Research 1, 73–83.

R. A. Ulichney, (1987), *Digital Halftoning*, The MIT Press, London.

S. Voß, (1993), *Solving Quadratic Assignment Problems Using the Reverse Elimination Method*, report 6/93 (revision march 1994), FB 1 / FG Operations Research, Technische Hochschule Darmstadt, Germany.

M. R. Wilhelm and T. L. Ward, (1987), *Solving quadratic assignment problems by simulated annealing*, IEEE Transactions 19, 107−119.

D. L. Woodruff and E. Zemel, (1993), *Hashing vectors for tabu search*, Annals of Operations Research 41, 123–137.

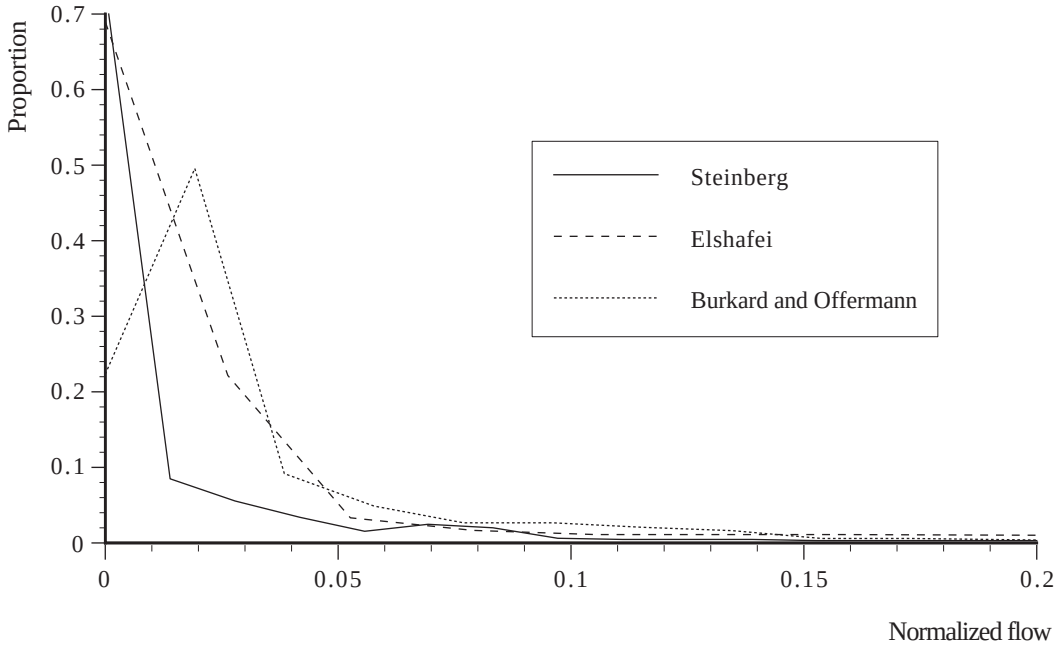

Figure 1 : Distributions of the flows for various real-life problems.

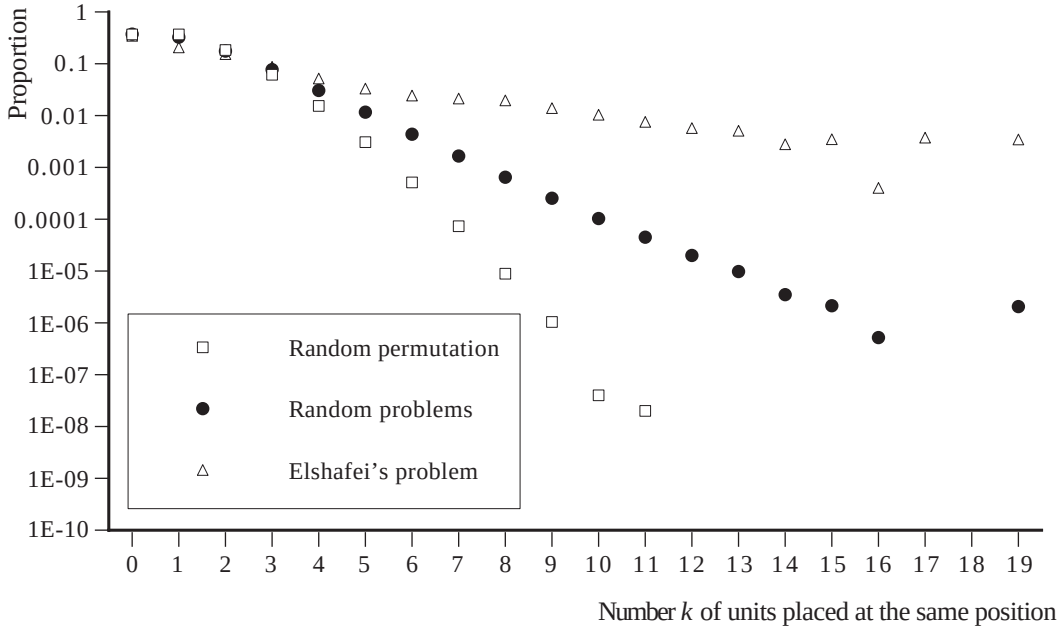

Figure 2 : Proportion of units placed at given positions.

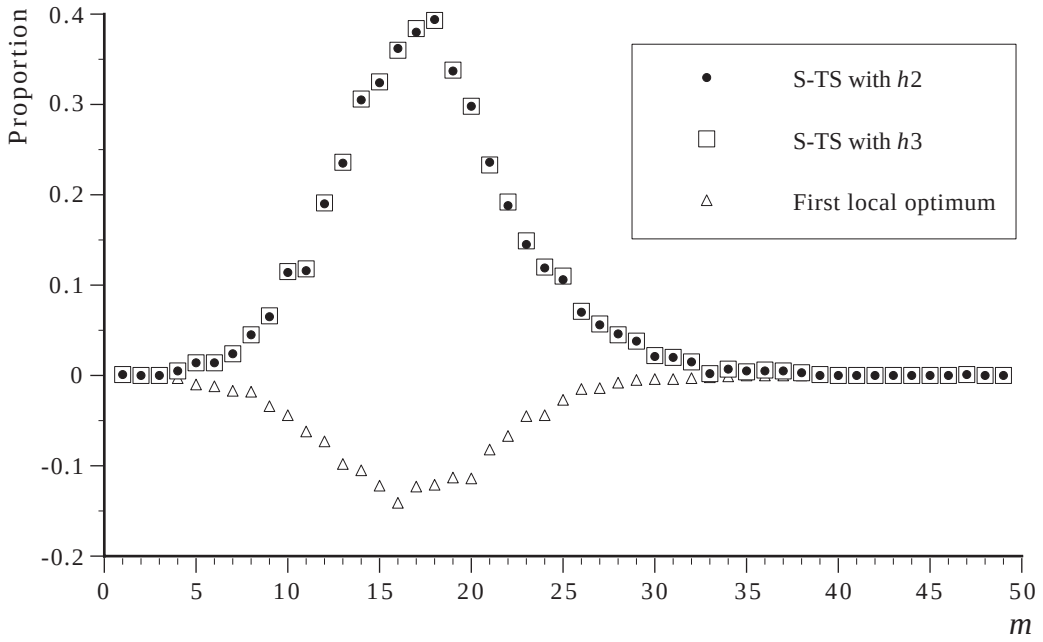

Figure 3 : Proportion of searches, regarding to Ro-TS, refinding a good solution to Tai50a.

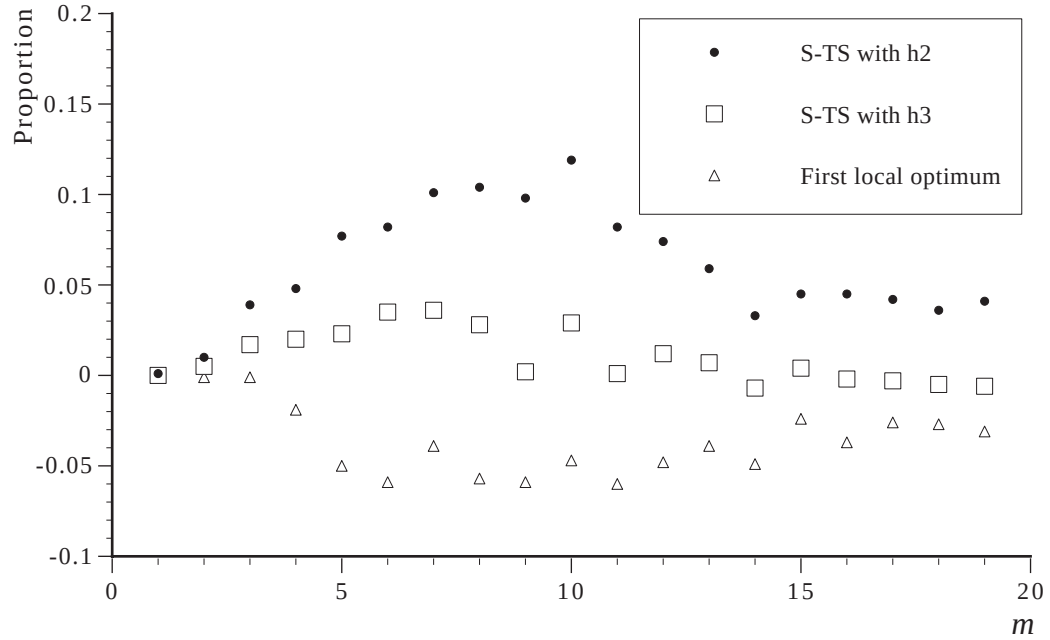

Figure 4 : Proportion of searches, regarding to Ro-TS, refinding a good solution to Nug20.

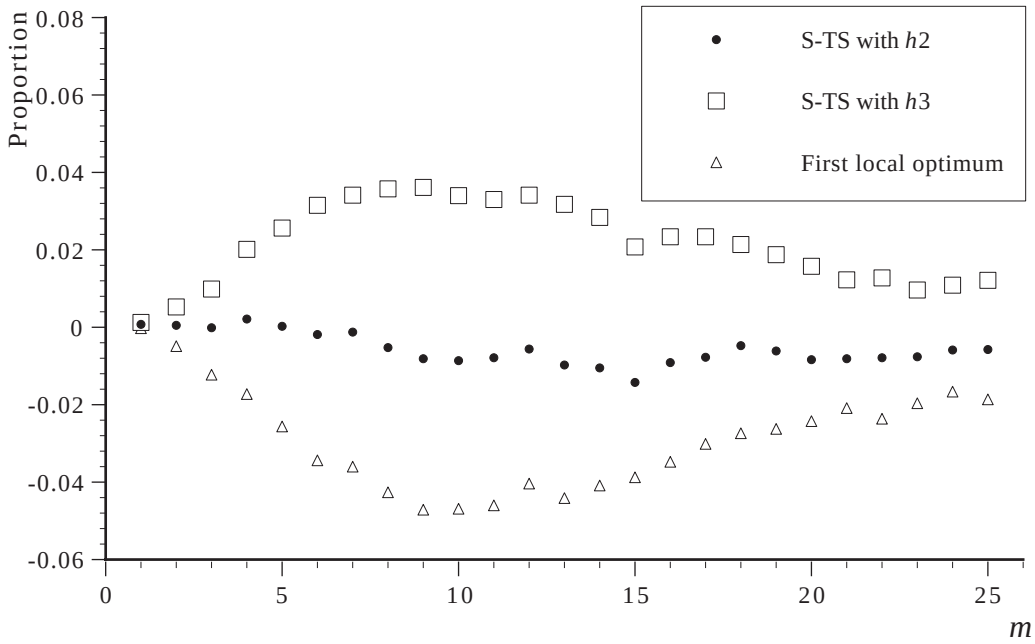

Figure 5 : Proportion of searches, regarding to Ro-TS, refinding goods solutions to Bur26a-h.

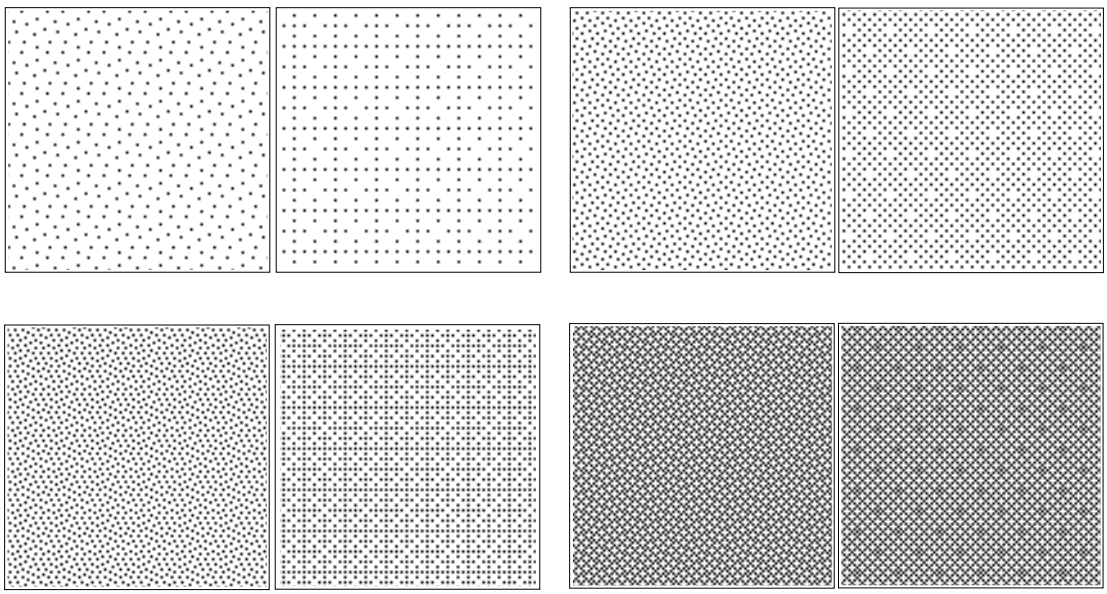

Figure 6 : Greys frames of densities 11/256, 30/256, 43/256 and 98/256 obtained by solving quadratic assignment problems (left) and usually used frames (right).

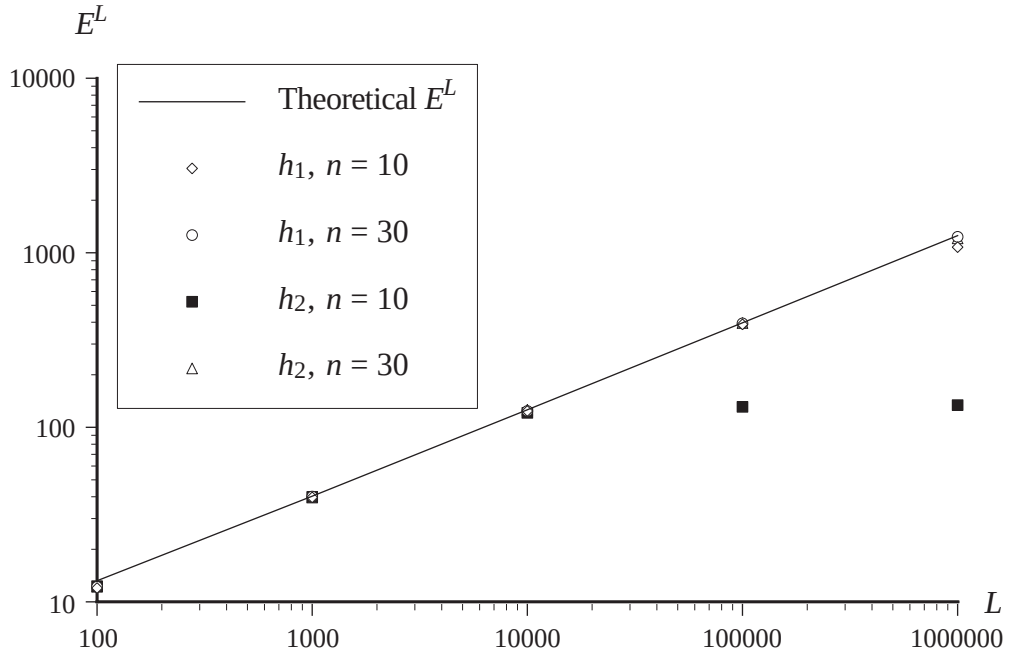

Figure 7 : Empirical and theoretical expectation E^L as a function of L.

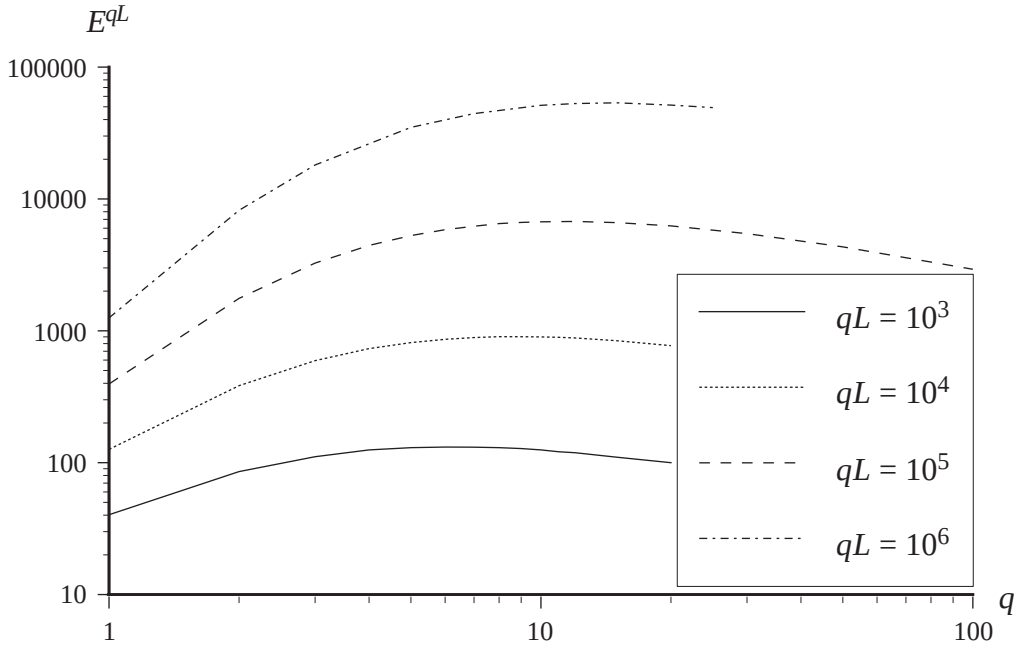

Figure 8 : Statistical expectation E^{qL} as a function of q for various total memory sizes qL.