Annals of Operations Research 139, 65–94, 2005

Recent Advances for the Quadratic Assignment Problem with Special Emphasis on Instances that are Difficult for Meta-Heuristic Methods

ZVI DREZNER zdrezner@fullerton.edu
California State University-Fullerton

PETER M. HAHN * hahn@seas.upenn.edu
The University of Pennsylvania

ÉRIC D. TAILLARD eric.taillard@eivd.ch
The University of Applied Sciences of Western Switzerland

Abstract. This paper reports heuristic and exact solution advances for the Quadratic Assignment Problem (QAP). QAP instances most often discussed in the literature are relatively well solved by heuristic approaches. Indeed, solutions at a fraction of one percent from the best known solution values are rapidly found by most heuristic methods. Exact methods are not able to prove optimality for these instances as soon as the problem size approaches 30 to 40. This article presents new QAP instances that are ill conditioned for many metaheuristic-based methods. However, these new instances are shown to be solved relatively well by some exact methods, since problem instances up to a size of 75 have been exactly solved.

Keywords: quadratic assignment problem, local search, branch & bound, benchmarks

1. Introduction

1.1. The quadratic assignment problem (QAP)

The QAP is a combinatorial optimization problem stated for the first time by Koopmans and Beckmann in 1957. It can be described as follows: Given two $n \times n$ matrices (a_{ij}) and (b_{kl}), find a permutation π minimizing:

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

We denote by $\Pi(n)$ the set of permutations of n elements. Sahni and Gonzalez (1976) showed that the QAP is *NP*-hard and that there is no ε-approximation polynomial algorithm for the QAP unless $P = NP$.

*Corresponding author.

One of the oldest applications of the QAP is the placement of electronic modules (Steinberg 1961; Nugent, Vollman, and Ruml, 1968). In this case, a_{ij} is the number of connections between electronic module i and module j and b_{kl} is the distance between locations k and l on which modules can be placed. By solving a QAP, one tries to minimize the total length of the electrical connections.

Another application is the assignment of specialized rooms in a building (Elshafei, 1977). In this case, a_{ij} is the flow of people that must go from service i to service j and b_{ij} is the time for going from room i to room j. A more recent application is the assignment of gates to airplanes in an airport; in this case, a_{ij} is the number of passengers going from airplane i to airplane j (a special "airplane" is the main entrance of the airport) and b_{kl} is the walking distance between gates k and l.

Finally, let us mention other applications in imagery (Taillard, 1995), turbine runner balancing (Laporte and Mercure, 1988) and the fact that problems such as the traveling salesman or the linear ordering can be formulated as special QAPs.

1.2. Literature review

The quadratic assignment problem was one of the first problems solved by metaheuristic methods first conceived in the 1980's. Burkard and Rendl (1984) proposed a simulated annealing procedure that was able to find much better solutions than all the previously designed heuristic methods. Six years later, Connolly (1990) proposed an improved annealing scheme. His method is easy to set up, since the user has to select the number of iterations; and all other parameters are automatically computed. The code for this method is available on the Internet at `http://ina.eivd.ch/collaborateurs/etd/default.html`. Thus, it was convenient for us to include Connolly's method in our computational results. At around the same time, Skorin-Kapov (1990) proposed a tabu search. Then, Taillard (1991) proposed a more robust tabu search, with fewer parameters and running n times faster than the previous implementation. Even though Taillard's method was proposed over 12 years ago, it remains one of the most efficient for some problem instances. It is available on the web (see address above). Other tabu searches have been proposed, such as the reactive tabu search of Battiti and Tecchiolli (1994a), the star-shape diversification approach of Sondergeld and Voss (1996), and the concentric tabu search of Drezner (2002). Recent improved tabu search algorithms are suggested in Misevicius (2003a, 2005). Battiti and Tecchioli (1994b) compared tabu search techniques with simulated annealing.

Genetic algorithms have also been proposed, for instance by Tate and Smith (1995), but hybrid approaches, such as those of Fleurent and Ferland (1994), Ahuja, Orlin, and Tiwari (2000), Drezner (2003, 2005) are more efficient. Another heuristic, GRASP (greedy randomized adaptive search procedure) was proposed by Li, Pardalos, and Resende (1994). More recently, ant system approaches (Gambardella, Taillard, and Dorigo, 1999; Stützle and Hoos, 1999; Taillard, 1998) have been proposed, as well as a scatter search (Cung et al., 1997). Some of these methods have been compared in Taillard (1995) who

showed that the efficiency of these methods strongly depends on the problem instance type to which they are applied.

While great progress has been made on generating good solutions to large and difficult QAP instances, this has not been the case for finding exact solutions. The history of solving QAP instances exactly centers on the now famous Nugent problems. In 1968, Nugent, Vollmann and Ruml posed a set of problem instances of size 5, 6, 7, 8, 12, 15, 20 and 30, noted for their difficulty. In these instances, the distance matrix stems from an $n_1 \times n_2$ grid with Manhattan distances between grid points. Most of the resulting QAP instances have multiple global optima (at least four if $n_1 \neq n_2$ and at least eight if $n_1 = n_2$). Even worse, these globally optimal solutions are at the maximally possible distance from other globally optimal solutions. At the time of their paper, it was an achievement to find the optimum solution to the size 8 Nugent instance. Enumerating all 8! (eight factorial) possible assignments took 3374 seconds on a GE 265 computer. It was not until Burkard solved the Nug8 in 0.426 seconds on a CDC Cyber76 machine (Burkard, 1975) that notable progress was made toward exact solution methods. For this, Burkard used Gilmore-Lawler (GL) lower bounds (Gilmore, 1962) in a branch and bound algorithm.

In the 1970's and 80's, one could only expect to solve difficult instances for $n < 16$. In 1978 Burkard and Stratmann optimally solved the Nug12 in 29.325 seconds on a CDC Cyber76 machine. They used a branch-and-bound perturbation algorithm; again utilizing the popular GL lower bound. The GL was to remain the preferred lower bounding technique until well into the 1990's. In 1980, Burkard and Derigs reported that they were the first to solve the Nug15 using a branch-and-bound code. They did this in 2947.32 seconds on a CDC Cyber76.

It was not until 1997 that Clausen and Perregaard were able to solve exactly the very difficult Nug20 instance. For this, they used a parallel branch-and-bound algorithm on a MEIKO computing surface with 16 Intel i860 processors. The wall time for solving the Nug20 problem was 57,963 seconds and required the evaluation of 360,148,026 nodes. On a single processor, the runtime would have been 811,440 seconds. Again, they relied upon the GL bound that had been developed decades earlier. Much progress has been made since then. Still, the exact solving of the most difficult of the Nugent instances, the Nug30 requires resources not commonly found at today's universities.

In the late 1990's two developments resulted in a large improvement in the ability to solve QAPs exactly. The first is a level-1 reformulation linearization technique (RLT) bound of Hahn and Grant (1998) and Hahn, Grant, and Hall (1998) that derives from a dual ascent procedure discovered much earlier (Hahn, 1968). Using this bound, Hahn, Hightower, Johnson, Guignard-Spielberg and Roucairol (2001) reported the general solution of the Nug25 in 1998. The second development is the quadratic programming bound (QPB) of Anstreicher and Brixius (2001a). These two methods respectively made it possible to solve exactly heretofore-unsolved problems of size 30, i.e., the Kra30a (Hahn and Krarup, 2001) and the Nug30 (Anstreicher, et al., 2002)

Hahn, with ideas from Hightower and Johnson, solved the Kra30a in April 1999. Anstreicher, Brixius, Goux and Linderoth solved the Nug30 in June 2000. Figure 1 shows the progress made in solving the Nugent instances.

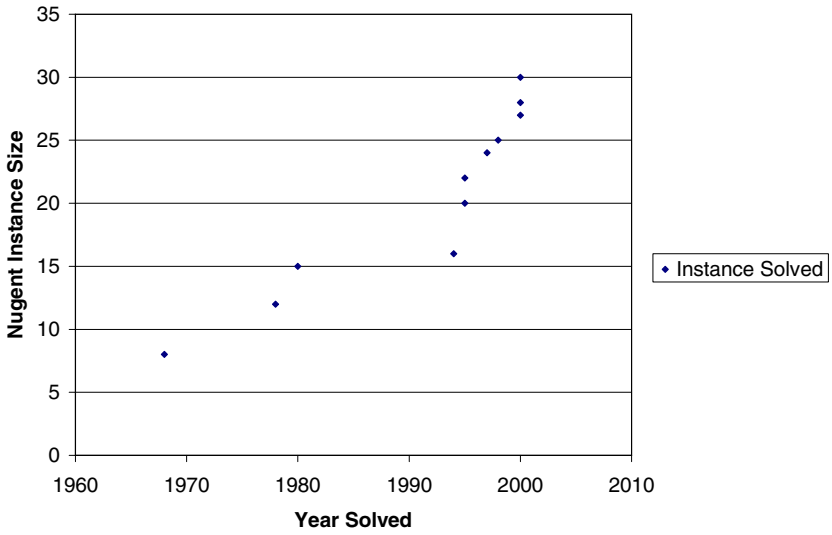

Figure 1. Nugent instances first solved.

This past year, Hightower and Hahn (Adams et al., 2003) developed a new lower bound for the Quadratic Assignment Problem (QAP) based on a level-2 reformulation-linearization technique (RLT). Their bound is based on a formulation of the QAP similar to the one developed by Ramachandran and Pekny in 1996. The new technique has its primary roots in the level-2 RLT concept of Sherali and Adams (1990 and 1994). The method of calculation of this new level-2 RLT bound is a generalization of the dual ascent procedure developed for the level-1 RLT bound calculation (Hahn and Grant, 1998).

The QAPLIB library (Burkard et al., 1997) available on the web at `http://www.seas.upenn.edu/qaplib/` contains about 140 different problem instances that are frequently used by researchers for comparing methods. The size of these instances ranges from 12 to 256, but the size of only 4 instances is larger than 100. Most of these instances are relatively well solved by heuristic methods (for instance, a 12 year old tabu search (Taillard, 1991) coded in a few dozen lines): almost all metaheuristic-based methods are able to find solutions at a fraction of one percent above the best known solutions. Exact methods have difficulties in proving optimality of the best solutions known for the problem instances contained in QAPLIB. Some instances of size lower than 30 are still open and none of the instances of size 40 or higher are exactly solved (except those with optimum known by construction).

One contribution of this study is to propose new problem instances whose characteristics are complementary to QAPLIB ones, namely, they are relatively well solved by exact methods but ill conditioned for some local search methods. These instances were proposed recently, so that no references for them are available. Thus, they are explained herein.

For researchers working on exact methods, these new instances are interesting, since they have been found to be solvable for sizes larger than 70. The main difficulty in dealing with such instances is the memory needed to store data structures. This kind of problem does not often arise with QAPLIB instances, since instances of size 40 cannot yet be solved exactly. In fact, a QAPLIB instance of size 25 (the Tai25a) was solved for the first time only recently, as reported in the QAPLIB (see http://www.seas.upenn.edu/qaplib/).

For researchers working on heuristic methods, these new instances are interesting for several reasons. First, most recent heuristic methods are built on neighborhood search based on transpositions (pair exchanges). This neighborhood concept is not effective for our new instances. Indeed, there are local optima with a value of the objective function more than 400% above the optimum. Therefore, a new neighborhood concept must be used. Second, the variance of the solutions produced by heuristic methods can be very high. It is not judicious to proceed to statistical comparisons by considering only average and standard deviation of the solution values produced by different methods, as is often the case in the literature. Therefore, new comparison techniques have to be constructed, and the present paper discusses a new way to compare iterative methods. Third, it is necessary to find faster heuristic methods, since the size of some of these new instances is quite large (up to 729).

The paper is organized as follows. In Section 2 we introduce two new classes of QAP problems which are difficult for metaheuristics and relatively easy for exact algorithms. In Section 3 two of the metaheuristics used for comparison are described. In Section 4 various exact methods are described, and in Section 5 we detail the statistical technique used for comparison among various heuristic methods. In Section 6 we present computational experiments with the new problems, and we conclude in Section 7.

2. New difficult instances

2.1. The Drexx instances

The new Dre (*xx* denotes size) problem instances are based on a rectangular grid, where all non-adjacent nodes have zero weight. This way, a pair exchange of the optimal permutation will result in many adjacent pairs becoming non-adjacent. Thus, the value of the objective function will increase quite steeply. The neighborhood of the optimum consists of solutions with much higher values of the objective function. Therefore, it is difficult to reach the optimum from a solution in the neighborhood of the optimum, but not in the neighborhood itself. These problem instances are available in `http://business.fullerton.edu/zdrezner`. The problems are generated by the following principles.

1. All problems are symmetric.
2. A configuration of k by l squares is the base for the problem.
3. The number of facilities is $n = k \times l$.

4. Each square has between two and four "adjacent" squares.
5. A random permutation (the optimal solution) is generated and facilities are "assigned"to the appropriate square by the permutation.
6. Most pairs of facilities get a zero weight (i.e., flow between them). Weight of flow is bi-directional. Only adjacent facilities get a randomly generated positive weight between one and ten.
7. Distances are generated such that all adjacent cells get a distance of one, and all other pairs of cells get a distance randomly generated between two and ten.

Note that:

1. The value of the objective function for the permutation is the sum of all the weights because for the permutation all positive weights are multiplied by a distance of "1".
2. The sum of the weights is also a lower bound because all distances are ≥ 1.
3. The permutation and three other mirror images for $k \neq l$ are the only optimal solutions.

There have been attempts to express the difficulty of QAP problem instances by measures that are relatively easy to compute. The best known is the *flow dominance* (i.e., the standard deviation of the flows, normalized by the average flow, or: $100 \times \max\{\text{std\_deviation}(a_{ij})/\text{average}(a_{ij}), \text{std\_deviation}(b_{ij})/\text{average}(b_{ij})\}$. Another measure is the landscape ruggedness (Angel and Zissimopoulos, 2001), defined as the standard deviation of the difference of two neighboring solutions. This standard deviation is also normalized, so that the ruggedness is a number between 0 and 100. The neighborhood considered for the QAP is the transposition of two elements. It is believed that the higher the flow dominance and ruggedness, the harder the problem is for local searches. For problem instance Dre90, the flow dominance is 554 (which is a very high value compared to other QAPLIB instances) and ruggedness is 99.7 (which is also very high).

2.2. *The Taixxeyy instances*

Construction of another set of problems that are difficult for heuristic methods is based on the observation that most available heuristic methods depend on a transposition neighborhood (i.e., exchange of two facilities). First, we searched for a problem structure for which local optima (for a transposition neighborhood) are relatively far apart. Second, we wanted very good solutions to be available, without necessarily knowing the global optimum, so that these problems would be attractive to researchers. Third, we wanted to be able to generate large problem instances. Finally, we wanted the difficulty of the instances to grow with size, especially for global search heuristic methods, such as genetic (hybrid) algorithms.

To meet the last condition, we observed that genetic algorithms, as well as exact methods, generally have difficulties in finding the optimum of problem instances for

which distances and flows are uniformly distributed (for example, the instance Tai25a, of size $n = 25$, mentioned above). However, such instances are relatively well approached by heuristic methods, i.e., solutions at a fraction of a percent above the best known solution are very easily found. In our new instances, we made sure that distances and flows would not be uniformly distributed.

We built our new instances recursively. For generating a QAP instance of size $st \times st$, we uniformly generated t matrices of size $s \times s$ of flows and distances. The t flow matrices were placed in a block-diagonal matrix of size $st \times st$, the elements outside the blocks were set to 0 and the t distance matrices were used to create a block-diagonal matrix, but with elements outside the blocks being set to infinity. These new $st \times st$ QAP instances can be solved as follows: First, consider the t^2 QAP instances of size $s \times s$ that can be obtained by combining each $s \times s$ flow matrix and each $s \times s$ distance matrix. These t^2 instances are optimally solved, and the optimum solution values are stored in a $t \times t$ matrix C. The optimal solution value of the $st \times st$ QAP instance can be obtained by solving the linear assignment problem on the C matrix.

In order to somewhat hide the block-diagonal structure of the $st \times st$ instance, a few elements outside the diagonal of the flow matrix are set to a small positive value and the elements outside the diagonal of the distance matrix are set to relatively large values, but not infinity. By doing that, the optimality property of the linear assignment problem on the C matrix is lost. Naturally, the process can be iterated and u $st \times st$ instances can be generated to create a new $stu \times stu$ instance. The Tai*xx*e*yy* problem instances (*xx* stands for the size of the instance and *yy* for the number of the instance) were created this way. Twenty instances of sizes $n = 27(= stu = 3 \cdot 3 \cdot 3), n = 45(= 5 \cdot 3 \cdot 3), n = 75(= 5 \cdot 5 \cdot 3), n = 125(= 5 \cdot 5 \cdot 5), n = 175(= 7 \cdot 5 \cdot 5), n = 343(= 7 \cdot 7 \cdot 7)$ and 10 instances of size $n = 729(= 9 \cdot 9 \cdot 9)$ are available on the web page: `http://ina.eivd.ch/collaborateurs/etd/problemes.dir/qap.dir/qap.html`

Figure 2 illustrates the distance structures of these instances when translated to the assignment of gates to airplanes. Let us suppose that an airport is composed of $u = 3$ terminals, each of them separated in $t = 3$ branches that have $s = 3$ gates each. In this case, the distances between the gates belonging to the same branch are very low. The distances between gates belonging to different branches of the same terminal are relatively large and the distance to travel from one terminal to another is much larger. For these instances, the flow matrix is strongly negatively correlated with the distance matrix, meaning that there are *tu* groups of *s* airplanes for which a large number of passengers have to transit into the group, while very few of them have to transit between different groups.

Since these instances embed uniformly generated instances, they are asymptotically difficult to solve exactly. For the case of a transposition neighborhood, one must move through *s* very bad solutions to make even a small change in the structure of a solution (exchanging two primitive blocks of size *s*) and *st* moves are required to exchange two secondary blocks. As explained below, tabu search or simulated annealing methods based on transposition neighborhood become inefficient with such a structure.

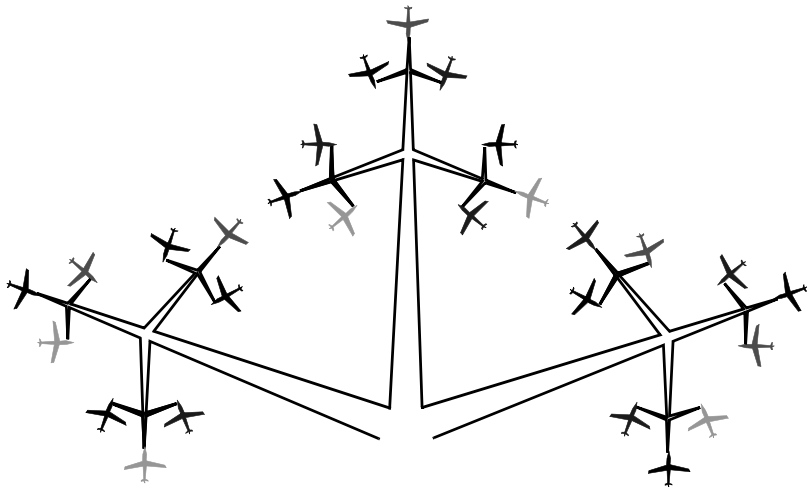

Figure 2. Typical configuration of a large airport. The distance matrices of Taixxeyy instances are related to such configurations, with different number of terminal, branches per terminal and gates per branch.

The flow dominance of Tai*xxeyy* is relatively low, around 100. For instance, the value for Tai343e01 is 104. The ruggedness is high, very close to the maximum value (99.9 for Tai343e01). Therefore, it can be deduced that the flow dominance is not a good measure for the difficulty of a problem instance.

3. Heuristic techniques considered

In this section, we briefly describe a number of heuristic methods designed for the quadratic assignment problem. First, we present the main ideas of the simulated annealing of Connolly (1990), then the "robust tabu search" of Taillard (1991) and the "reactive tabu search" of Battitti and Tecchiolli (1994a). Then, we describe the hybrid-genetic algorithms of Fleurent and Ferland (1994) and Drezner (2005). An implementation of a generalization of hybrid-genetic algorithms, scatter search, developed by Cung et al. (1998) is then presented. We end this brief description of heuristic methods with the Fast Ant system (Taillard, 1998).

3.1. Simulated annealing

Simulated annealing is a meta-heuristic based on the definition of a neighborhood. For the QAP, almost all implementations use a transposition neighborhood (i.e. exchanging the assignment of 2 elements). In order to not be trapped in a local optimum, relative to the neighborhood structure considered, the basic idea of simulated annealing is: at each iteration, to randomly choose a neighboring solution and to accept the move to

this new solution if it is better than the current solution. If the neighboring solution is poorer than the current solution by an amount Δ, the neighboring solution is accepted with probability $e^{-\Delta/T}$, where T is a parameter, called the temperature. The temperature is changed (typically decreased) every iteration. The set of temperatures is called the cooling scheme. The contribution of Connolly (1990) is a proposed cooling scheme that is automatically adapted to the problem instance and to the number of iterations specified by the user.

3.2. Tabu searches

Tabu search is also a neighborhood-based method. To our knowledge, all tabu searches for the QAP use the transposition neighborhood (or extensions of it, Drezner(2002)). The basic idea of tabu search is at each iteration to choose the best neighbor, even if the best neighbor is worse than the current solution. To avoid cycling by re-visiting the same subset of solutions (e.g. to go back to the same local optimum just after having left it), a data structure, called the tabu list is implemented. This list memorizes solutions (or attributes of solutions) that are "forbidden" (may not be chosen) for a number of iterations. The contributions of Taillard (1991) with his *robust tabu search* are: 1. an $O(n^2)$ procedure to examine the whole neighborhood (the complexity of earlier implementations was $O(n^3)$), 2. a random, automatic policy for choosing the number of interdictions during which attributes of solutions are forbidden and 3. a long term diversification mechanism. The contribution of Battiti and Tecchiolli (1994) with the *reactive tabu search* is to automatically adapt the number of iterations during which solutions are forbidden. This number is increased if the same solution is visited twice and it is decreased when no cycles occur. When the search stagnates, an escape mechanism that performs random moves is activated.

3.3. Hybrid genetic algorithms

The idea of genetic algorithms is to maintain a population of solutions. Two solutions of the population are selected for creating a new individual. The creation process consists of mixing the components of both individuals with a *crossover* operator and then to slightly and randomly modify the new solution with a *mutation* operator. The new mutated solution is then incorporated into the population. Finally, to limit population size, some solutions are periodically eliminated from the population. Tate and Smith (1995) have proposed such an algorithm for the QAP, but its efficiency is rather low. The main idea of Fleurent and Ferland (1994) is to use Tate and Smith's crossover operator, but to replace the mutation operator by a procedure that improves the solution with a local search. In the numerical results that follow, we have considered a population of size $P = 100$ and a mutation operator that performs $4n$ iterations of the robust tabu search.

The hybrid genetic algorithm described in Drezner (2003, 2005) is one of the most powerful hybrid-genetic algorithms to-date. Comparisons between this hybrid genetic

```
1 A starting population is randomly selected, and the post-
  merging procedure is applied on each starting population
  member.
2 The following is repeated for a pre-specified number of
  generations:
  2a  Two population members are randomly selected (each
      population member has an equal chance of being selected,
      regardless of its objective function value) and merged to
      produce an offspring.
  2b  The post merging procedure is applied on the merged
      solution, possibly improving it.
  2c  If the value of the objective function for the offspring
      is not better than the value of the objective function for
      the worst population member, the offspring is ignored and
      the process of the next generation starts.
  2d  Otherwise,
            If the offspring is identical to an existing
            population member, it is ignored and the process of
            the next generation starts.
            If the offspring is different from all population
            members, the offspring replaces the worst
            population
            member.
```

Figure 3. General scheme of the hybrid genetic algorithm.

algorithm and the genetic algorithm in Ahuja, Orlin, and Tewari (2000) demonstrated the superiority of this hybrid genetic algorithm (Drezner, 2003, 2005).

A genetic algorithm produces offspring by mating parents and attempting to improve the population make-up by replacing existing population members with superior offspring. A hybrid genetic algorithm, sometimes called a memetic algorithm (Moscato, 2002), applies an improvement heuristic (a post merging procedure) to every offspring, before considering its inclusion into the population. The specific hybrid genetic algorithm tested in this paper is presented in Figure 3.

For the implementation of the hybrid genetic algorithm for the solution of the QAP, each solution is defined by the list of facilities assigned to sites $1, \ldots, n$. The merging procedure used is the "cohesive merging procedure" and the post-merging procedure used is the concentric short tabu search. These are described in detail in Drezner (2003, 2005).

To implement the general scheme of Figure 3, we employ the compounded genetic algorithm proposed in Drezner (2005). For Step 1 of Figure 3, we run the short search with zero levels ten times with a population size of $P = 100$ and $\max\{2000, 40n\}$ generations. We collect the best ten population members at the termination of each run and create a starting population of $P = 100$. For Step 2, we use the short search with 6 levels and $\max\{1000, 20n\}$ generations.

The computational complexity of one iteration is $O(n^3)$. The complexity of the algorithm depends on the expected number of iterations and the number of generations.

1) $I = \emptyset$, $J = \emptyset$
2) While $|I| < n$ repeat:
2a) Choose i, randomly, uniformly, $1 \leq i \leq n$, $i \notin I$.
2b) Choose j, randomly, $1 \leq j \leq n$, $j \notin J$, with probability $\tau_{ij} \Big/ \sum_{1 \leq k \leq n,\, k \notin J} \tau_{ik}$ and set $\mu_i = j$.
2c) $I = I \cup \{i\}$, $J = J \cup \{j\}$

Figure 4. Construction of a provisory solution.

3.4. *Scatter search*

Scatter search (Glover, 1977) can be viewed as a generalization of genetic algorithms. It also uses a population of solutions. But, during the process, the population is controlled with clustering techniques in order to maintain the solutions' diversity while keeping solutions of high quality. Indeed, one of the weaknesses of genetic algorithms is that the population tends to contain solutions that are very similar to one another. In our implementation of the Fleurent and Ferland genetic hybrid algorithm, at the end of the process the population consisted only of replications of one solution. Scatter search also produces new solutions by mixing the properties of solutions of the population. But, the number of solutions that can participate in the creation of new solutions is not limited to two, as is found in genetic algorithms. In the Cung et al. (1997) implementation, new solutions are improved with robust tabu search.

3.5. *Fast ant system* (*FANT*)

The concept behind the ant system is to attribute an "interest" for each component of a solution. One part of this interest can be computed *a priori* (e.g., for the traveling salesman problem, the a priori interest to perform a travel between two cities is inversely proportional to their distance) and another part is computed *a posteriori*, when the process has produced and evaluated solutions. This a posteriori interest is called pheromone trace level in ant system metaphor (Colorni, Dorigo, and Maniezzo, 1992). In ant systems, solutions are built randomly, the probability of choosing a component being proportional to its interest. For the QAP, several ant systems have been designed. Most of them do not consider an a priori interest and the a posteriori interest is implemented as a matrix T of size $n \times n$, whose entry τ_{ij} is a statistic over the setting of $\pi_i = j$ in previously constructed solutions π.

In FANT (Taillard, 1998), all entries of T are set to 1, initially. For generating a new (provisory) solution μ, a constructive method that chooses the elements of μ successively, in a random order and with a probability proportional to the values contained in the T matrix, is used. More formally, the constructive method is presented in figure 4:

The provisory solution μ generated at the first step is generally not very good because, in the first iteration, μ is just a random permutation. Therefore, the first steps of

1) For $i = 1$ to n do:
  1a) $\tau_{i\pi_i} = \tau_{i\pi_i} + r$
  1b) $\tau_{i\pi_i^*} = \tau_{i\pi_i^*} + r^*$

Figure 5. Update of trace matrix T.

a first improving neighbor procedure are applied to μ. This procedure can be executed in $O(n^3)$; it does not necessarily return a local optimum, but it is fast and may produce different solutions when starting with the same initial, not locally optimal solution. The improved solution is called π.

The statistic stored in T is based on two parameters, r and r^*, that represent the reinforcement of the matrix entries corresponding to the solution π and respectively π^*, the best solution produced by the algorithm. During the entire process, r^* is a constant parameter set by the user while r may vary. Initially $r = 1$ and $\tau_{ij} = r$, where $1 \leq i$ and $j \leq n$, meaning that the memory does not initially contain any information. Usually, the entries of matrix T are updated as indicated in figure 5:

Where π is the solution produced at the current iteration and π^* is the best solution produced so far. In two cases, this update is done in another way:

(1) If π^* has been improved, then r is re-set to 1 and all the entries of T are set to 1. The aim of this re-setting is to intensify the search around π^* by giving less importance to the past of the search.

(2) If the provisory solution μ generated at step 2b is equal to π^*, then r is incremented by one unit and all the entries of T are set to r. This situation occurs when π^* has not been improved for a large number of iterations, meaning that the re-enforcement of the entries corresponding to π^* is too high. The aim of this strategy is to diversify the search when the information contained in T is not much different from π^*.

In addition to the number of iterations to perform (= number of solutions built by artificial ants), the method has only one parameter, r^*.

4. Exact methods

In this section we compare the three most promising exact solution techniques for the QAP. The first is the branch-and-bound algorithm based upon the level-1 RLT bound

The RLT is a general theory for reformulating mixed 0-1 linear and polynomial programs in higher-variable spaces in such a manner that tight polyhedral outer-approximations of the convex hull of solutions are obtained. The RLT was first introduced in Adams and Sherali (1986, 1990) and is best described in Sherali and Adams (1990). The technique is ideally suited to calculating lower bounds for 0–1 quadratic programs such as the QAP. Level-1 of the RLT provides the weakest linear programming approximation of the RLT hierarchy. Nevertheless, it provides one of the strongest lower bounds

that have been developed for QAP branch and bound enumeration purposes. The level-1 RLT multiplies every constraint by each binary variable x and enforces $x^2 = x$. It also enforces the assignment constraints. To complete the linearization, a linear variable is substituted for each quadratic term in the objective function and in the constraints.

The second promising exact solution algorithm for the QAP is the branch-and-bound algorithm of Anstreicher and Brixius (2001a). Their algorithm uses a convex quadratic programming relaxation to obtain a bound at each node. Their bound (QPB) is based on the projected eigenvalue bound of Hadley, Rendl, and Wolkowicz (1992) and uses a semi-definite programming characterization of a basic eigenvalue bound from Anstreicher and Wolkowicz (1998). In their branch and bound algorithm, the QPB is computed approximately using the Frank-Wolfe algorithm (FW) (Anstreicher and Brixius, 2001b). FW can be run for any number of iterations, producing a bound and a matrix of dual variables. The branch and bound algorithm makes extensive use of the dual matrix in the branching step.

The third promising branch and bound algorithm for the QAP is based on the level-2 RLT lower bound calculation of Hightower and Hahn (Adams et al., 2003). Level-2 RLT is the second level in the RLT hierarchy. The method is similar to the level-1 RLT case with two modifications. It multiplies every constraint by each individual variable x and each product of variables $x_i x_j$ in the reformulation and enforces $x^2 = x$. It then linearizes by substituting a variable for each quadratic and each cubic term. The result is a bound that is at least as strong as the level-1 bound.

Anstreicher and Brixius reported groundbreaking computational results on the Nugent QAPs. For instance, on the Nugent 25 instance they achieved branch-and-bound enumeration evaluating 80, 430, 341 nodes in a wall time of 6.7 hours on a Master-Worker (MW) distributed processing system developed as part of the MetaNEOS project at Argonne National Labs. During this 6.7 hours, the number of active worker machines averaged 94. The equivalent computation time on a single HP C3000 workstation would be approximately 13 days. Whereas, the earlier level-1 RLT branch-and-bound enumeration required longer runtime (52 days on a single CPU Ultra 10 with a 360 MHz processor), though it required examination of half as many (38, 726, 326) nodes.

Anstreicher and Brixius constructed Nugent 27 and 28 instances by removing the end rows and columns of the flow matrix of the Nugent 30. They used a 3 by 9 grid for the Nugent 27 distance matrix and a 4 by 7 grid for the Nugent 28 distance matrix. They are the first to have solved exactly these two instances. The Nugent 27 was first solved in a wall time of approximately 24 hours, during which the number of worker machines averaged 136 and peaked at 238. The equivalent computation time on a single HP C3000 workstation would be approximately 126 days. It required the evaluation of 513,160,139 nodes. The Nugent 28 was first solved in 4 days, 8 hours wall time. The number of active worker machines averaged approximately 200. The equivalent runtime on a single HP C3000 workstation would be 435 days. Solving the Nugent 28 required the evaluation of 2, 935, 394, 013 nodes.

For large QAP instances that are difficult for exact methods, such as the Nugent instances, the level-2 RLT bound gives shorter run times than either the level-1 RLT

Table 1
Comparison of Competing Branch-and-Bound Algorithms (times normalized to an HP C3000 cpu).

Instance	Measurement	Level-1 RLT	QPB	Level-2 RLT
Nug 27	No. of Nodes	297,648,966	∼402,000,000	46,315
	Norm minutes	458,923	94,608	37,639
Nug 28	No. of Nodes	N/A	∼2,230,000,000	202,295
	Norm minutes	N/A	462,528	198,674
Kra 30a	No. of Nodes	29,764,589	N/A	17,193
	Norm minutes	99,371	N/A	38,652
Kra 30b	No. of Nodes	183,659,980	5,136,036,412	N/A
	Norm minutes	367,200	1,403,352	N/A
Nug 30	No. of Nodes	N/A	11,892,208,412	543,061
	Norm minutes	N/A	3,647,664	1,369,692
Ste 36a	No. of Nodes	5,225,559	∼1,790,000,000	N/A
	Norm minutes	27,649	36,540	N/A

N/A = Not available.

bound or the QPB (see Hahn et al., 2002). For instance, the Nugent 25 was solved using the level-2 bound in the equivalent of 9.8 days on an HP C3000 workstation and required the evaluation of only 30,718 nodes. The reason for the shorter time is that this bound is so much tighter. It takes longer to compute, but this doesn't matter, since far fewer nodes need to be evaluated. Interestingly, attempting first to fathom nodes with a level-1 RLT bound and following that with level-2 RLT bound when necessary, results in even better performance. With this strategy, the Nugent 25 is solved in 5.9 days on an HP C3000 workstation and requires the evaluation of only 16, 649 nodes.

Table 1 summarizes the performance of these three exact solution algorithms for six difficult QAP instances, sizes 27 to 36. The times in Table 1 are normalized to the speed of a single HP C3000 CPU. The QPB performance figures in Table 1 are taken from (Anstreicher et al., 2002) and show improvements in the QPB branch and bound results reported two paragraphs before this one. For the Nug27, the QPB outperforms the Level-1 RLT by about 4.8-to-1, whereas for the Kra30b the Level-1 RLT outperforms the QPB by about 2.6-to-1. For the Nug27, the Level-2 RLT outperforms the QPB by 2.5-to-1 and for the Nug28 it outperforms the QPB by 2.3-to-1. For the Nug30, Level-2 RLT is faster than QPB by 2.66-to-1. Level-2 RLT shows additional promise, since the number of nodes evaluated is very small. Innovative techniques for cutting down computational effort could reduce runtimes even further.

5. Comparing non-deterministic heuristic algorithms

A general method for comparing non-deterministic iterative searches is presented. This method follows the lines of Taillard (2001 and 2002).

Comparing two (or more) heuristic methods based on metaheuristic principles is a difficult task that has not been solved satisfactorily. Heuristic solution methods are usually

iterative, meaning that the longer they run, the better are the solutions they produce. They are also almost all non-deterministic, since they generally depend on a pseudo-random number generator. This means that it is possible to obtain two different solutions when running the same heuristic method twice.

It is clear that comparing iterative methods must be done considering both the quality of the solution produced and the computational effort. The last is traditionally the computing time on a given machine. Unfortunately, this measure is both imprecise and volatile. Indeed, computing time depends on the programming style, on the compiler, on the compiling options, etc. Moreover, the lifetime of computer equipment is very limited. Sometimes, the computer is already obsolete when the articles describing a new method are published.

If the combinatorial problem consists of optimizing a unique objective function, it is easy to compare the quality of two solutions. However, the comparison of non-deterministic methods is not so easy. The solution quality not only depends on the computational effort but also on the pseudo-random number generator seed. Suppose that heuristic method A is compared to heuristic method B and A and B were run n_A and n_B times, respectively. During these runs, each improvement of solution quality is recorded with the corresponding computational effort such that, for a given time t, the quality of the $n_A + n_B$ solutions produced by both methods is known. We wish to determine whether A is better than B at time t.

The solution quality obtained by methods A and B, respectively, at time t are random variables $X_{A(t)}$ and $X_{B(t)}$, respectively. The probability density functions of these random variables are $f_{A(t)}$ and $f_{B(t)}$, respectively. We wish to compare the mathematical expectations $\mathrm{E}(X_{A(t)})$ and $\mathrm{E}(X_{B(t)})$ in order to know which one is lower and decide which method is better at time t. Unfortunately, $f_{A(t)}$ and $f_{B(t)}$ are unknown. Moreover, the number of runs n_A and n_B are typically limited to a few dozen. Therefore, it is not tractable to attempt the determination of the density functions $f_{A(t)}$ and $f_{B(t)}$ with the data available. We cannot assume (as is often done in literature) that these curves are normally distributed (for instance, these functions are bounded by the value of the global optimum while normal distributions are unbounded). Therefore, a comparison of both methods on the basis of average and standard deviation of solution values observed is generally not appropriate.

Other hypothesis tests have to be used. The Mann-Whitney test allows a proper comparison. Translated to the comparison of a non-deterministic heuristic method, the null hypothesis is $f_{A(t)} = f_{B(t)}$. If the probability that this hypothesis is true is lower than a given confidence level α, considering the n_A and n_B solutions obtained, then the null hypothesis is rejected and the alternative hypothesis is accepted, (i.e., the probability that A produces a better solution than B is larger (or lower) than the probability that B produces a better solution than A).

In short, the Mann-Whitney test is performed as follows: first, the $n_A + n_B$ solutions are ranked by decreasing quality and a rank between 1 and $n_A + n_B$ is attributed to each solution. If many solutions have the same value of the objective function, they all get the same rank, i.e., the average of the rank they would have obtained if there were no

equality. Then, a statistic $S_{A(t)}$ is computed. This statistic corresponds to the sum of the ranks of solutions issued from method A. If $S_{A(t)} > T_\alpha(n_A, n_B)$, then the null hypothesis is rejected with a confidence level of α and it is accepted that A is worse than B. The values of $T_\alpha(n_A, n_B)$ can be found in tables. There are also books that provide tables that give α as a function of $S_{A(t)}$ and $n_A + n_B$. More details on the Mann-Whitney test can be found in Conover (1999).

This test has to be repeated for different values of computational time t. A convenient way to compare methods A and B is to graphically draw the probability that A is better than B as a function of the computational effort. Naturally, such a comparison is possible only if reliable computational times can be obtained. However, it could happen that one method could be implemented in such a way that it runs faster. Therefore, we have to be particularly careful when comparing computing times, for instance, by considering a multiplicative safety factor ϕ in the measure of computing times.

The computing times are an essential criterion when comparing heuristic methods, but this criterion is unreliable. If one wants to be independent of the computer used for presenting the comparisons, it is required to consider an absolute computational effort, such as the number of iterations. When one can derive the mathematical complexity for a single iteration, a more reliable computational measure can always be obtained.

6. Computational experiments with the new problems

In this section we report a few computational results obtained with our new problem instances. The goal of this section is not to show that one method is superior to all the others, but to illustrate the difficulty of the task of comparing iterative methods. We report the results of comparisons between the robust tabu search (Taillard, 1991), the compounded hybrid genetic (Drezner, 2005), the simulated annealing of Connolly (1990), and the FANT described in Section 3.5. In Section 6.1 we compare the two tabu searches. In Section 6.2 we compare the hybrid genetic algorithms and the scatter search algorithm. In Section 6.3 we report computational experience with solving the new problems by exact methods.

6.1. Computational experience with tabu search

In order to illustrate the difficulty of comparing different methods, let us start with the comparison of two methods that are relatively similar, the robust tabu search of Taillard (1991) and the reactive tabu search of Battiti and Tecchiolli (1994). Figure 6 illustrates the difference of behavior of both tabu search methods on the new problem instance Tai27e01 (optimum value 2,558). This comparison is done along the lines suggested in Section 5. When considering only the average solution values obtained over 20 independent runs of both methods, it could be (erroneously) deduced that reactive tabu is much better than robust tabu for computational effort lower than 2,000 iterations. Between 2,500 and

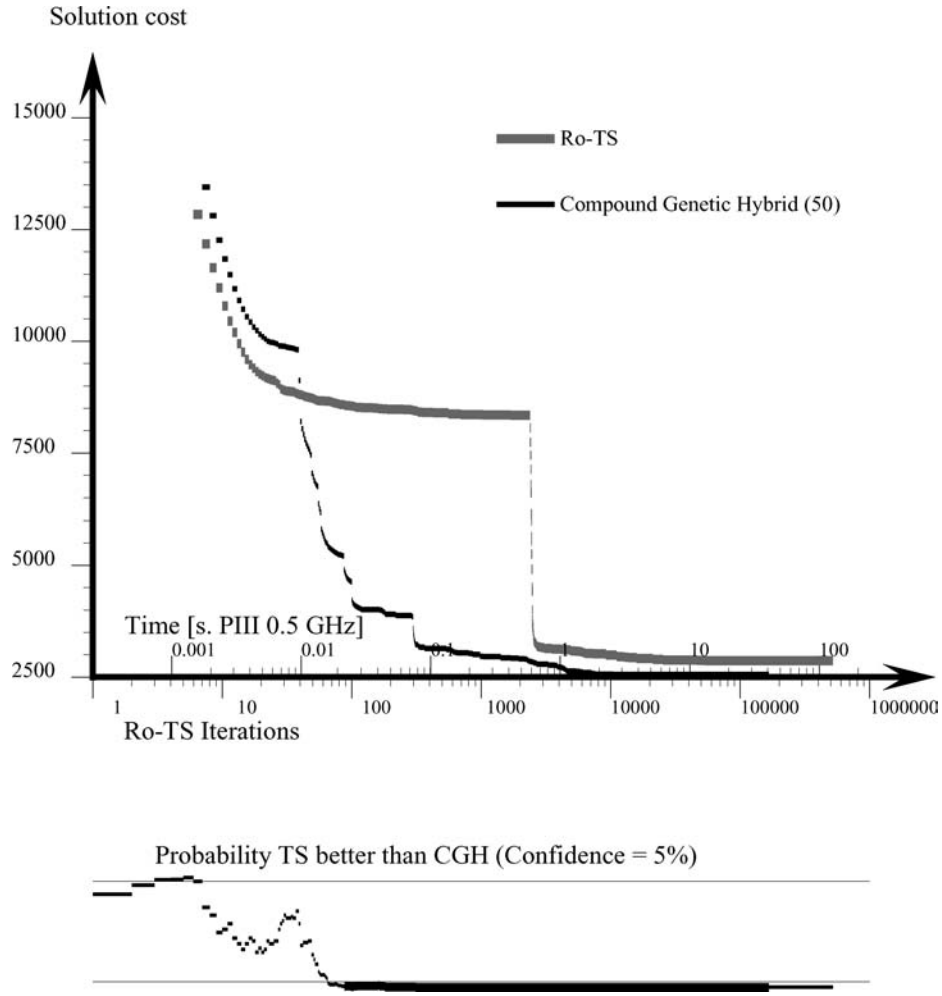

Figure 6. Comparison between robust tabu search and reactive tabu search on Tai27e01 instance.

25,000 iterations robust tabu is much better and, after 80,000 iterations, reactive tabu might be slightly better.

The variance of the solution value (not shown in Figure 6 for clarity reasons) is very high (except for large computational efforts, when almost all runs have found the optimum). Therefore, a classical (parametric) statistical test for comparing the average of both methods cannot show the superiority of a method, while the non-parametric test we used is often able to show it. Therefore, we note that the second (confidence) diagram contains a lot of information. Indeed, it answers the most basic and interesting question: Is method A significantly better than method B after computational effort t?

When looking at this confidence diagram, reactive tabu is significantly superior to robust tabu only for a very narrow interval around 1,500 iterations. Even if reactive

tabu is on the average more than 100% above robust tabu for 4,000 iterations, this is not significant. Figure 6 also reveals that an early termination of both methods can lead to very bad solutions and that the diversification mechanism of robust tabu started earlier (after 2,200 iterations) than the escape mechanism of reactive tabu. The number of iterations needed by tabu searches for producing solutions of acceptable quality strongly depends on the diversification mechanism. The Tai*xxeyy* instances are especially ill conditioned for tabu searches. As shown further, the small Tai*xxeyy* instances are better solved with genetic hybrids or scatter search algorithms.

6.2. Computational experience with genetic hybrids and scatter search

Genetic hybrids and scatter search are methods that present similarities, such as the use of a population of solutions and operators for combining solutions. In this section, we compare implementations of 1. the genetic hybrid of Fleurent and Ferland (1994), programmed in C/C++ and run on a Pentium III 500 MHz, 2. genetic hybrids of Drezner (2003, 2005), programmed in FORTRAN and run on a 600 MHz Toshiba Portege 7200 laptop, and 3. the parallel scatter search of Cung et al. (1998), run on a single Pentium III 800 MHz processor with time-sharing parallel processes (Cung and Donadio, 2002). Due to the use of different machines, different programming languages and different management systems, it is perilous to try to compare these methods using computational times. Fortunately, all these methods are hybridized with tabu searches and most of the computational effort is spent in tabu iterations. So, it is quite pertinent to measure the computational *effort* in terms of *number of tabu iterations*, as suggested in Section 5.

6.2.1. Results for Drexx

In Figure 7, we compare the genetic hybrid short search with $P = 100$ GHS(100) with the genetic hybrid (GH) of Fleurent and Ferland (1994) for problem instance Dre30. GH also uses a population of 100 solutions. We see in this figure that, between a total of 300 and 7,000 tabu search iterations, GHS(100) is superior to GH. At 10,000 iterations, GH starts to use efficiently the information contained in the population of solution and beats GHS(100). The bold line in the Confidence diagram at about 10s indicates that GH would remain significantly better than GHS(100) even if GH runs $\phi = 2$ times slower. After 100,000 iterations, GH has converged and does not improve the best solution anymore. Since GHS eliminates duplicated solutions in the population, the convergence is slower but allows finding better solutions.

Figure 8 compares the genetic hybrid short search of Drezner (2003, 2005) with the parallel version of the scatter search of Cung et al. (1997) for problem instance Dre30. The numerical results of scatter search come from Cung and Donadio (2002). In this figure, we see that scatter search requires a relatively long time before producing the first solution. Then, it clearly beats GHS.

In Table 2, we report computational results obtained by the compounded hybrid genetic algorithm for different Dre*xx* instances. Each instance was solved 20 times. We

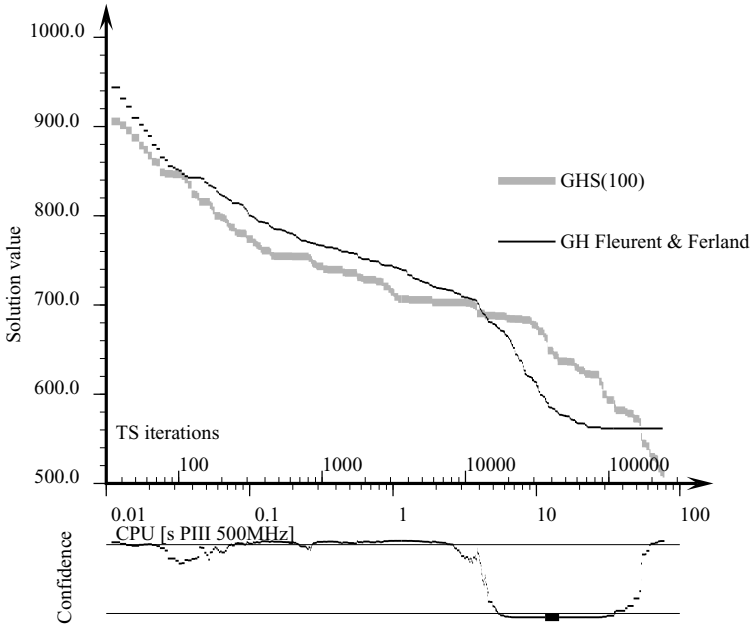

Figure 7. Comparison of the genetic hybrid short search GHS(100) (Drezner 2003, 2005c) with the genetic hybrid GH of Fleurent and Ferland (1994), for problem instance Dre30.

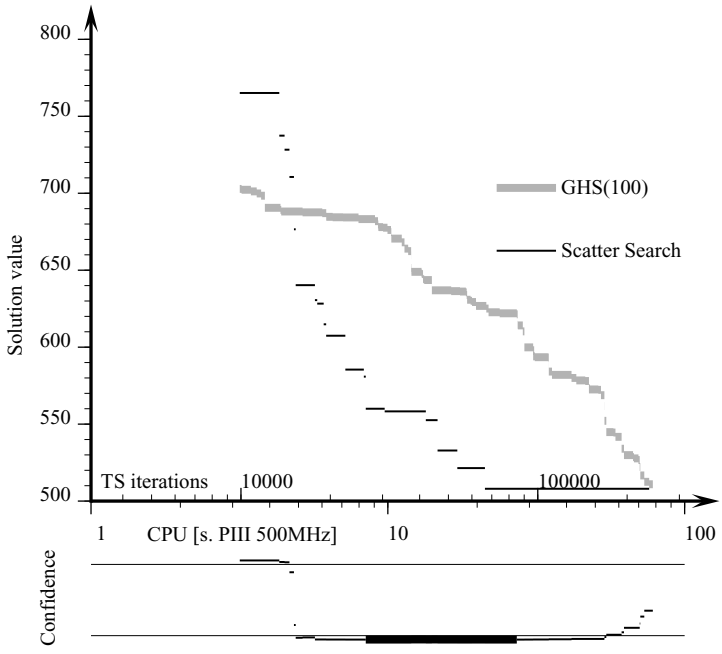

Figure 8. Comparison of Genetic hybrid short search, GHS (Drezner 2003, 2005) with parallel scatter search (Cung and Donadio 2002) for problem instance Dre30.

Table 2
Results for hybrid genetic on Drexx instances.

		Minimum (20 runs)			Average		
Problem	Opt.	Value	%/Opt.	# Times	Value	%/Opt.	Time (min.)
Dre30	508	508	0	20	508.0	0	2.39
Dre42	764	764	0	18	774.2	1.34	9.13
Dre56	1086	1086	0	6	1275.2	17.46	30.18
Dre72	1452	1452	0	2	1848.1	27.28	93.19
Dre90	1838	2218	20.67	0	2460.7	33.88	192.63

recorded the minimum obtained, the number of times out of 20 that the minimum (or optimum) is obtained, the average percentage over the best-known (or optimal) solution, and the run time in minutes per run. Parameter settings are given in Section 3.3. Table 2 provides best and average solution values (absolute and relative to the optimum). Examining this table, we conclude that the difficulty of the instances rapidly grows with their size. Solving Dre90 required over 3 hours of computer time for each replication. The experimental time complexity of our method grows as $O(n^4)$.

The results in Table 2 are quite good. However, when compared with other instances of the QAPLIB of similar sizes (see Drezner, 2005) it is clear that these instances are much more difficult. The best solution found for Dre90 was more than 20% over the optimum! For example, the six Sko100 problems were solved in Drezner (2005) in about 160 minutes, found the best-known solution 85% of the time (and found it at least 9 times out of 20 replications for all 6 problems). And, the average over the best-known solution was 0.003% and never exceeded 0.007%. We conclude that the Dre*xx* problems are much more difficult for the hybrid genetic algorithm than are the Sko100 problems.

6.2.2. Results for Taixxeyy

GHS (Drezner, 2005) found the best solution known for all Tai27e*yy* and Tai45e*yy* problems in all 20 replications. The run times are about 1 and 5 minutes per run, respectively, on the average with very small variation. For comparison, we tested this hybrid genetic algorithm (the simple, not the compounded version) on the Nug30 problem and it solved the Nug30 problem in about 18 seconds per replication and found the optimum in *all* 1,000 replications.

Run times for Tai75e*yy* are about 37 minutes per run. The genetic hybrid performed relatively well on the Tai75eyy instances, finding the best known solution 18.85 times out of 20 replications for each of the 20 instances (377 times out of a total of 400 replications). The percentage of the average result over the best known solution ranged from 0, for the 11 problems that found the best known solution in all 20 replications, to 0.339% for Tai75e02 (for which the best known solution was found only 14 times out of 20 replications). The average percentage over the best-known solution for all 20 instances was 0.056%, which is much higher than is reported in Drezner (2005) for

Table 3
Results for hybrid genetic of Fleurent and Ferland on few Taixx instances.

Name	Best value known	Average %over best	Std deviation (%)	Average nr generations	Total tabu iterations
tai27e01	2558	0	0	218	23539
Tai45e01	6412	4.48	4.3	352	63335
Tai75e01	14488	10.1	2.9	704	211085
Tai125e01	35450	15.6	4.7	868	433953
Tai343e01	136288	18.0	2.2	1180	1619120

QAPLIB problems with a similar size. We conclude that the Taixxeyy are more difficult for the hybrid genetic algorithm than similar size problems listed in the QAPLIB. The computational times are relatively high (it took about 10 CPU days to gather all the results).

Therefore, we ran the faster genetic hybrid of Fleurent and Ferland on larger Tai*xx*e01 problem instances. Table 3 provides the main computational results obtained. We indicate in this table the value of the best solution value known (optimum for problem instances up to $n = 75$), the relative average solution value observed (expressed in % above best value known), the relative standard deviation, the average number of generations before convergence and total number of tabu search iterations performed up to convergence. In this Table, we see that the quality of the solutions obtained decreases with the size of the problem and the dispersion can be relatively high, even for small instances.

In Figure 9, the principles outlined in Section 5 have been used to compare the robust tabu search of Taillard (1991) with the genetic hybrid of Fleurent and Ferland (1994) on our new problem instance Tai27e01. We see in this figure that the solution values obtained at the beginning of the search are very bad for both methods. Robust tabu search (Ro-TS) is trapped for many iterations in bad local optima while the genetic hybrid finds the structure of good solutions earlier. A second diagram is depicted in this figure, plotting the probability of Ro-TS to be better than the hybrid genetic method. The horizontal lines in this diagram indicate the probabilities 0.05 and 0.95. We observe in the second diagram that the genetic hybrid method is significantly better than Ro-TS as soon as the computational effort reaches the equivalent of about 100 Ro-TS iterations. The genetic hybrid would remain significantly better than robust tabu, even if it runs $\phi = 2$ times slower (indicated in bold in the confidence diagram).

6.2.3. Comparing several heuristic methods

For comparing two methods, this second diagram is much more valuable than a classical table reporting average (and best, worst, standard deviation, etc.) of the solution values for a given computational effort, and is more compact. By putting several of such diagrams in the same figure, it is possible to compare many heuristic methods in one plot. This was done in figures 10 and 11, respectively, in which we compare five different methods, each

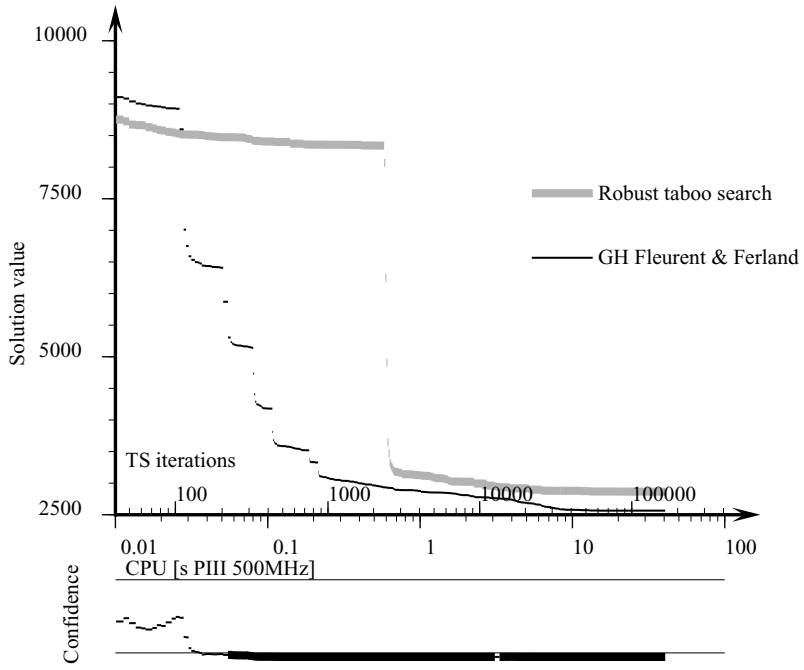

Figure 9. Comparison of robust tabu search with the genetic hybrid of Fleurent and Ferland on the new problem instance Tai27e01.

of them run 20 times, for problem instances Dre30 and Tai27e01, respectively. These methods are the simulated annealing of Connolly (1990), the tabu search of Taillard (1991), FANT with parameter $r^* = 5$, compounded hybrid genetic with parameter $P = 50$ and genetic short search with $P = 100$. From these two figures, we conclude that simulated annealing (SA) is significantly worse than the other methods (for these two problem instances and for almost every computational effort). We also see in this figure that there is no strict dominance of one method over the others. Depending on the computational time available and instance type, either method could be recommended.

6.3. Exact methods

While the new Dre*xx* and Tai*xxeyy* instances were indeed difficult for heuristic solution methods, it was not surprising to find them easy for exact methods. It was, however, surprising that these instances solved most easily using the level-1 RLT lower bound method of Hahn and Grant (1998). One would think that the level-2 RLT method would be better, since it produces tighter lower bounds.

Both the level-1 and level-2 RLT approaches were tried. But, it was determined early in the experimentation that the level-1 RLT gave sufficiently tight bounds at the

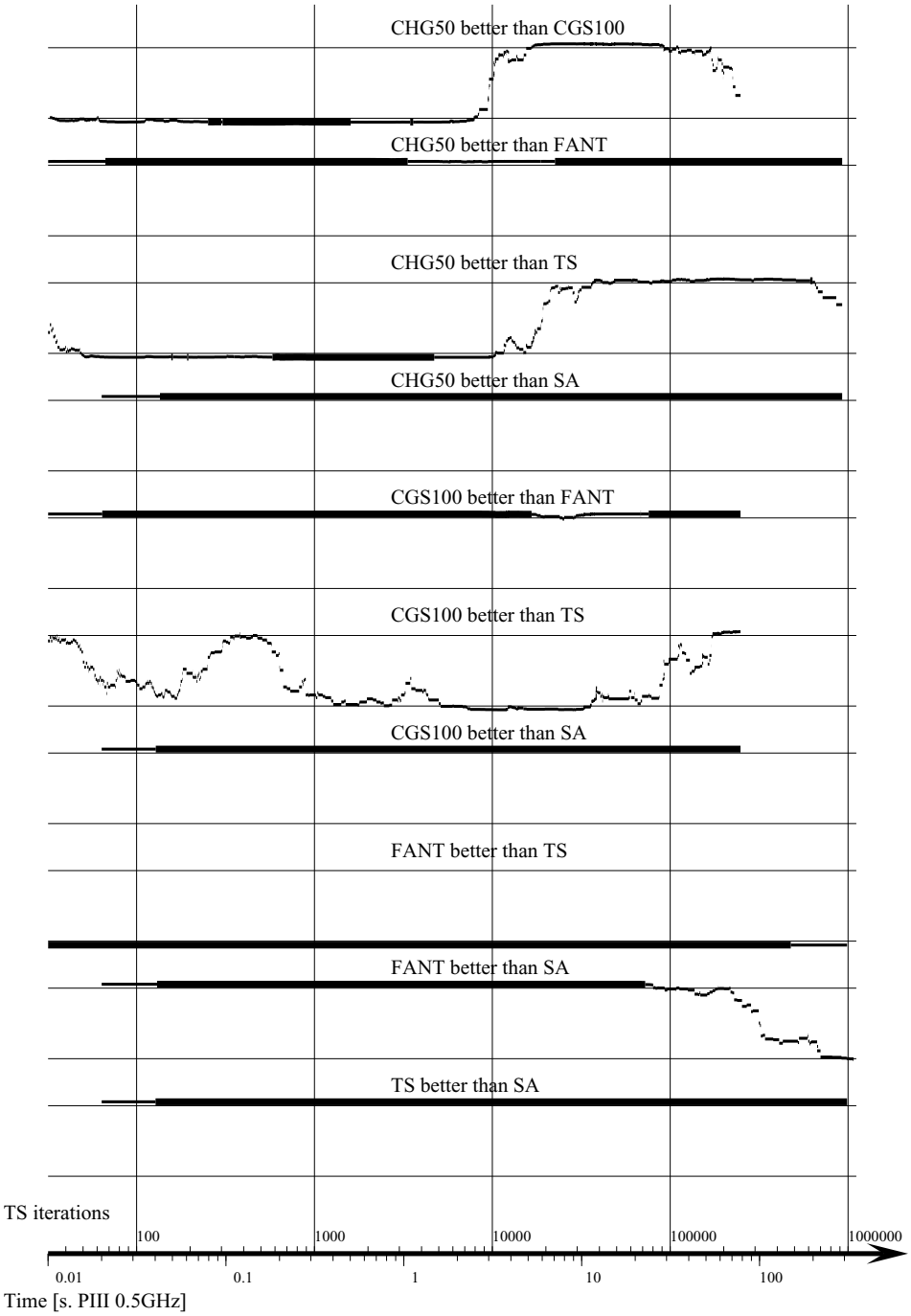

Figure 10. Probability diagrams (confidence: 5–95%) comparing five different methods all together on instance Dre30.

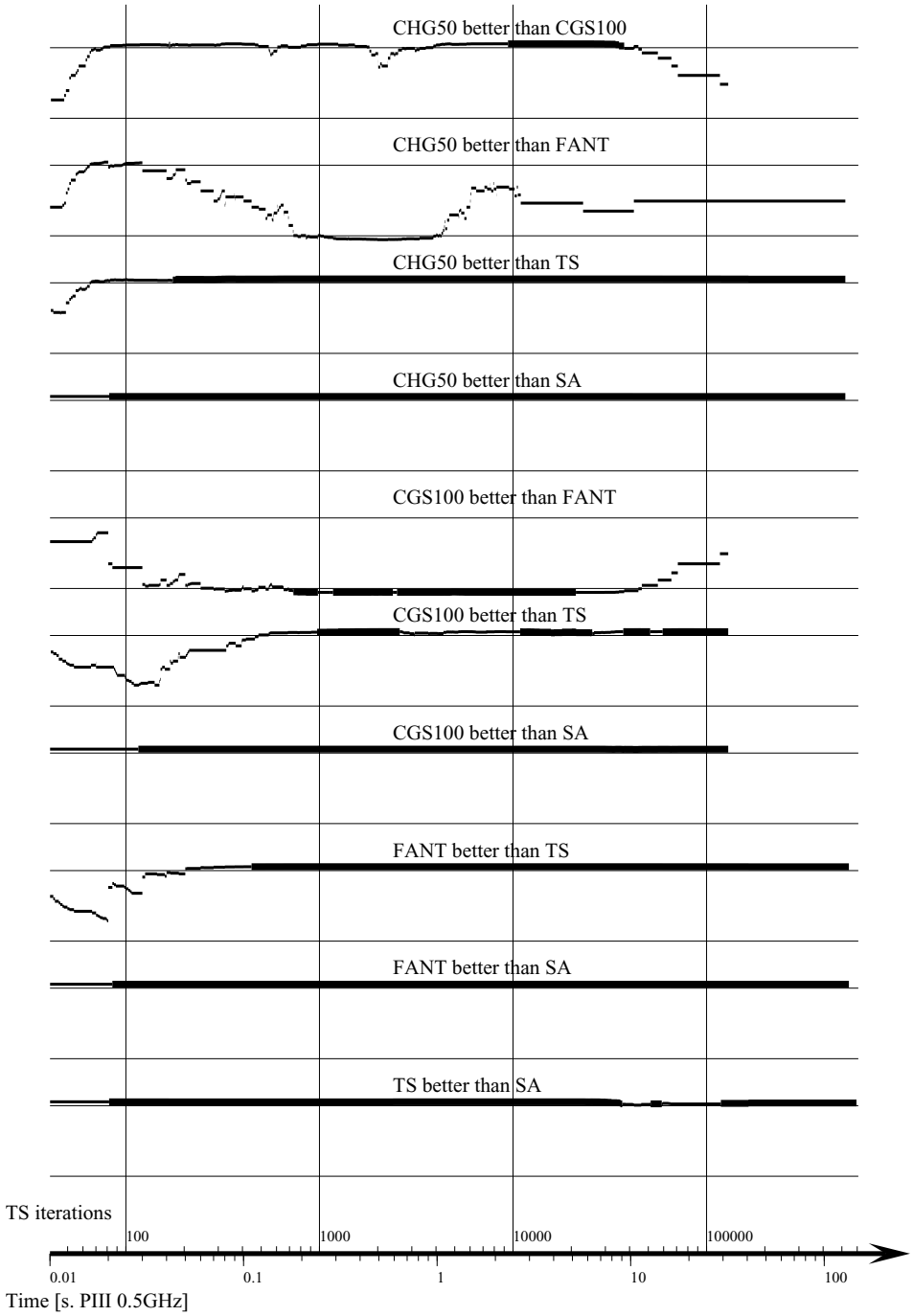

Figure 11. Probability diagrams (confidence: 5–95%) comparing five different methods all together on instance Tai27e01.

Table 4
Level-1 RLT Lower Bounds for Drexx Instances (time normalized to HP C3000 cpu-machines in Table 5).

		100 Iterations		500 Iterations	
Instance	Optimum	Normalized Time (secs)	Bound	Normalized Time (secs)	Bound
Dre18	332	4.5	331.1	22.3	331.5
Dre21	356	13.0	54.2	64.1	355.5
Dre24	396	19.0	393.4	89.7	394.9
Dre28	476	59.7	473.2	295.0	474.9
Dre30	508	72.2	503.7	356.5	506.1
Dre42	764	401.3	753.3	1,955.1	760.0
Dre56	1,086	386.9	1,055.1	1,885.3	1,069.7
Dre72	1,452	2,707.8	1,398.9	16,373.0	1,425.0

Table 5
Level-1 B & B Enumeration for Drexx Instances (time normalized to HP C3000 cpu—see speed factor.

Instance	No. of nodes	Normalized time(secs)	Memory (Mbytes)	Machine	Speed factor
Dre18	144	1.1	13	Sun Ultra 10	0.68
Dre21	409	3.9	20	Sun Ultra 10	0.68
Dre24	578	7.8	33	Sun Ultra 10	0.68
Dre28	909	31.7	69	Sun Ultra 10	0.68
Dre30	929	49.7	93	Sun Ultra 10	0.68
Dre42	2,165	496.9	417	HP J5000	1.40
Dre56	8,299	7,411.9	1,041	SGIO 2000 (UI)	0.55
Dre72	33,277	215,510.8	3,250	SGIO 2000 (UP)	0.63

root and was efficient for branch-and-bound enumeration. On the other hand, the level-2 RLT bounds, which were indeed tighter, took longer to calculate, so that the resulting enumeration runtimes were much longer. For instance, the Tai27e03 enumeration took 667.2 seconds using the level-2 RLT; whereas using the level-1 RLT on this instance took only 57.6 seconds. Thus, the results reported here are for only the level-1 RLT calculations. The results of the experiments using exact methods are presented in Tables 4 through 7.

Table 4 lists the lower bound calculations on the original problems for a selected set of the Dre*xx* instances. Table 5 lists the exact solution (branch-and-bound enumeration) runtimes and number of nodes evaluated for the same set of Drexx instances, as in Table 4. Table 6 lists the lower bound calculations and the exact solution runtimes and number of nodes evaluated for all 20 of the Tai27exx instances. All runs reported in Table 6 were made on a Sun Ultra 10 with a single 366 MHz processor. Table 7 lists the lower bound calculations and the exact solution runtimes and number of nodes evaluated for the Tai45e01 and Tai75e01 instances. In this table, the runtimes are normalized to the

Table 6
Lower bounds and B&B runtimes for Tai27exx instances.

		Lower bound at root						
Tai27		100 Iterations		500 or less iterations		Level 1 B&B enumeration**		
Instance	Optimum	Runtime (secs)*	Bound	Runtime (secs)*	Bond	No. of nodes	Runtime (secs)*	No. of optima
e01	2,558	26.0	2,554	Solved at root		1	40.1	n/a
e02	2,850	22.3	2,795	104.3	2,836	240	22.5	2
e03	3,258	23.8	3,257	46.2	3,258	207	57.6	2
e04	2,822	23.0	2,737	109.7	2,795	103	24.9	1
e05	3,074	23.5	3,044	45.8	3,074	553	94.1	4
e06	2,814	23.2	2,814	Solved at root		1	47.1	n/a
e07	3,428	23.3	3,354	110.2	3,422	78	13.4	1
e08	2,430	22.6	2,429	43.6	2,429	242	50.7	2
e09	2,902	22.3	2,796	104.9	2,838	53	25.5	1
e10	2,994	22.5	2,994	Solved at root		1	25.9	n/a
e11	2,906	22.3	2,895	Solved at root		1	28.8	n/a
e12	3,070	22.6	3,070	Solved at root		1	71.9	n/a
e13	2,966	22.3	2,825	105.0	2,872	333	63.2	2
e14	3,568	22.5	3,491	105.3	3,539	168	20.3	1
e15	2,628	22.2	2,603	104.5	2,608	323	51.9	2
e16	3,124	Solved at root		Solved at root		1	20.0	n/a
e17	3,840	22.3	3,514	105.2	3,600	718	115.2	1
e18	2,758	22.3	2,752	104.0	2,755	78	11.0	1
e19	2,514	22.3	2,495	Solved at root		1	49.9	n/a
e20	2,638	22.4	2,623	Solved at root		1	65.1	n/a

*Runtime on a Sun Ultra 10 (multiply runtime by 0.68 to get time on HP C3000.
**Requires 45 MB of RAM.
n/a = not available.

Table 7
Level-1 RLT runtimes for large Taillard instances (times normalized to HP C3000 cpu).

		Lower bound at root						
		100 Iterations		500 Iterations		B & B Enumeration		
Instance	Optimum	Time (secs)	Bound	Time (secs)	Bound	No. of nodes	Time (secs)	Memory (Mbytes)
Tai45e01	6,412	205.5	6,383	solved at root		1	361.5	498
Tai75e01	14,488	1,375.2	13,654	6,776	14,096	58,631	348,186	3,950

speed of a single HP C3000 processor. We are now solving Dre90 and it is expected to take about 75 days to solve on a single 733 MHz cpu of a Dell 7150 server.

7. Discussion

We presented herein a summary of the progress made in both heuristic and exact solutions for the QAP. To put this in perspective, we digressed to describe the historical context and the many contributions that led to the recent advances in QAP research. Much of the discussion was centered on problem instances found in the web-based repository for QAP data and results QAPLIB. However, early in the writing of this paper, it was deemed important to introduce and address a class of QAP problem instances that might arise in practice. These are instances specifically designed to be difficult for metaheuristic solution methods.

Accordingly, two types of quadratic assignment problems were generated that are difficult for common heuristic techniques. We, of course, tested these on the best metaheuristic methods available in order to prove and gauge their difficulty. To complete our work, we further tested these instances on some exact solution methods, expecting that they would remain difficult as well in that domain. But, it turned out that both types of instances are relatively easy for a modern exact algorithm. One of these exact algorithms solved many of these so-called "difficult" problems (with up to 75 facilities) in reasonable computer time.

We also proposed a statistical analysis to compare non-deterministic heuristic methods and prove statistically that one method is better than the other. This technique is universal and may prove very useful for comparisons of heuristic techniques for the solution of other problems as well.

Acknowledgments

Part of this research was conducted while the first author was visiting the Graduate School of Management, University of California at Irvine.

The work by the second author was supported in part by an international travel grant INT-9900376 from the National Science Foundation.

We thank Van Dat Cung and Sébastien Donadio for providing numerical results to us for scatter search.

References

Adams, W. et al. (2003). "A Lower Bound for the Quadratic Assignment Problem Based on a Level-2 Reformulation-Linearization Technique." OPIM Department Report No. 03-05-06, The Wharton School, University of Pennsylvania.

Adams, W. and H. Sherali. (1986). "A Tight Linearization and an Algorithm for Zero-One Quadratic Programming Problems." *Management Science*, 32(10), 1274–1290.

Adams, W. and H. Sherali. (1990). "Linearization Strategies for a Class of Zero-One Mixed Integer Programming Problems." *Operations Research* 38, 217–226.

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

Angel, E. and V. Zissimopoulos. (2001). "On the Landscape Ruggedness of the Quadratic Assignment Problem." *Theoretical Computer Science* 263, 159–172.

Anstreicher, K. and N.W. Brixius. (2001a). "A New Bound for the Quadratic Assignment Problem Based on Convex Quadratic Programming." *Mathematical Programming* 89, 341–357.

Anstreicher, K. and N.W. Brixius. (2001b). "Solving Quadratic Assignment Problems Using Convex Quadratic Programming Relaxations." *Optimization Methods and Software* 16, 49–68.

Anstreicher, K. et al. (2002). "Solving Large Quadratic Assignment Problems on Computational Grids." *Mathematical Programming* 91, 563–588.

Anstreicher, K. and H. Wolkowicz. (1998). "On Legrangian Relaxation of Quadratic Matrix Constraints." *Research Report CORR 98-24*. Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada.

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

Battiti, R. and G. Tecchiolli. (1994b). "Simulated Annealing and Tabu Search in the Long Run: A Comparison on QAP Tasks." *Computers and Mathematics with Applications* 28, 1–8.

Burkard, R. (1975). "Numerische Erfahungen Mit Summen-Und Bottleneck-Zuordnungsproblemen." In L. Collatz and H. Werner (eds.), *Numerische Methoden Bei Graphentheoretischen und Kombinatorischen Problemen.* Basel: Birkhauser Verlag.

Burkard, R. et al. (1997). "QAPLIB—A Quadratic Assignment Problem Library." *Journal of Global Optimization* 10, 391–403, electronic update: http://www.seas.upenn.edu/qaplib/ (revised 02.04.2003).

Burkard, R. and U. Derigs. (1980). "Assignment and Matching Problems." *Solution Methods with FORTRAN Programs*. New York: Springer-Verlag.

Burkard, R. and F. Rendl. (1984). "A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems." *European Journal of Operational Research* 17, 169–174.

Burkard, R. and K.H. Stratmann. (1978). "Numerical Investigations on Quadratic Assignment Problems." *Naval Research Logistical Quarterly* 25, 129–148.

Clausen, J. and M. Perregaard. (1997). "Solving Large Quadratic Assignment Problems in Parallel." *Computational Optimization and Applications* 8, 111–128.

Colorni, A., M. Dorigo and V. Maniezzo. (1992). "Distributed Optimization by Ant Colonies." In F.J. Varela and P. Bourgine (eds.) *Proceedings of ECAL'91—European Conference on Artificial Life*, MIT Press, Cambridge MA, 134–142.

Connolly, D.T. (1990). "An Improved Annealing Scheme for the QAP." *European Journal of Operational Research* 46, 93–100.

Conover, W.J. (1999). *Practical Nonparametric Statistics* (3rd Ed.). New York: Wiley Publishing Company.

Cung, Van Dat et al. (1997). "A Scatter Search Based Approach for the Quadratic Assignment Problem." In *Proceedings of the IEEE International Conference on Evolutionary Computation and Evolutionary Programming (ICEC'97),* Indianapolis, pp. 165–170.

Cung, Van Dat and S. Donadio. (2002). "Résultat du Scatter Search en Version Séquentielle et en Version Parallèle sur les Problèmes Dre30 et Tai27e01." Technical Report of OPALE Laboratory, France: Université de Versailles.

Dorigo, M. and L.M. Gambardella. (1997). "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem." *IEEE Transanctions on Evolutionary Computation* 1, 53–66.

Drezner, Z. (2002). "Heuristic Algorithms for the Solution of the Quadratic Assignment Problem." *Journal of Applied Mathematics and Decision Sciences* 6, 163–173.

Drezner, Z. (2003). "A New Genetic Algorithm for the Quadratic Assignment Problem." *INFORMS Journal on Computing* 15, 320–330.

Drezner, Z. (2005). "Compounded Genetic Algorithms for the Quadratic Assignment Problem." *Operations Research Letters* (in press).

Elshafei, A. (1977). "Hospital lay-out as a Quadratic Assignment Problem." *Operational Research Quarterly* 28, 167–179.

Fleurent, C. and J. Ferland. (1994). "Genetic Hybrids for the Quadratic Assignment Problem." *DIMACS Series in Math. Theoretical Computer Science* 16, 190–206.

Gambardella, L.M., E. Taillard and M. Dorigo. (1999). "Ant Colonies for the Quadratic Assignment Problem." *Journal of the Operational Research Society* 50, 167–176.

Gilmore, P. (1962). "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem." *Journal of the Society of Industrial and Applied Mathematics* 10, 305–313.

Glover, F. (1977). "Heuristics for Integer Programming Using Surrogate Constraints." *Decision Sciences* 8, 156–166.

Hadley, S.W., F. Rendl, and H. Wolkowicz. (1992). "A New Lower Bound Via Projection for the Quadratic Assignment Problem." *Mathematics of Operations Research* 17, 727–739.

Hahn, P.M. (1968). "Minimization of Cost in Assignment of Codes to Data Transmission." Ph.D. Dissertation, University of Pennsylvania, (1968). Available at: http://www.seas.upenn.edu/∼hahn/

Hahn, P.M. and T.L. Grant. (1998). "Lower Bounds for the Quadratic Assignment Problem Based Upon a Dual Formulation." *Operations Research* 46, 912–922.

Hahn, P.M., T.L. Grant and N. Hall. (1998). "A Branch-and-Bound Algorithm for the Quadratic Assignment Problem Based on the Hungarian Method." *European Journal of Operational Research* 108, 629–640.

Hahn, P.M. et al. (2001). "Tree Elaboration Strategies in Branch-and-Bound Algorithms for Solving the Quadratic Assignment Problem." *Yugoslav Journal of Operations Research* 11, 41–60.

Hahn, P.M. and J. Krarup. (2001). "A Hospital Facility Problem Finally Solved." *The Journal of Intelligent Manufacturing* 12, 487–496.

Hansen, P. and N. Mladenovic. (2001). "Variable Neighborhood Search: Principles and Applications." *European Journal of Operational Research* 130, 449–467.

Koopmans, T. and M.J. Beckmann. (1957). "Assignment Problems and the Location of Economics Activities." *Econometric* 25, 53–76.

Laporte, G. and H. Mercure. (1988). "Balancing Hydraulic Turbine Runners: A Quadratic Assignment Problem." *European Journal of Operational Research* 35, 378–381.

Li, Y., P.M. Pardalos, and M.G.C. Resende. (1994). "A Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. In P. Pardalos and H. Wolcowicz (eds.), *Quadratic Assignment and Related Problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 16, 237–261.

Misevicius, A. (2003a). "A Modified Tabu Search Algorithm for the Quadratic Assignment Problem." (under review).

Misevicius, A. (2005). "A Tabu Search Algorithm for the Quadratic Assignment Problem." *Computational Optimization and Applications*, 30, 95–111.

Mladenovic, N. and Pierre H. (1997). "Variable Neighborhood Search." *Computers and Operations Research* 24, 1097–1100.

Moscato, P. (2002). "Memetic Algorithms." In P.M. Pardalos and M.G.C. Resende (eds.), *Handbook of Applied Optimization*. Oxford, U.K. Oxford University Press.

Nugent, C., T. Vollman and J. Ruml. (1968). "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations." *Operations Research* 16, 150–173.

Ramachandran, B. and J.F. Pekny. (1996). "Dynamic Factorization Methods for Using Formulations Derived from Higher Order Lifting Techniques in the Solution of the Quadratic Assignment Problem." *State of the Art in Global Optimization: Computational Methods and Applications*. Dordrecht, Netherlands: Kluwer Academic Publishers, pp. 75–92.

Sahni, S. and T.F. Gonzalez. (1976). "P-Complete Approximation Problems." *Journal of the ACM* 23, 555–565.

Sherali, H.D. and W.P. Adams. (1990). "A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems." *SIAM Journal on Discrete Mathematics* 3, 411–430.

Sherali, H.D. and W.P. Adams. (1994). "A Hierarchy of Relaxations and Convex Hull Characterizations for Mixed-Integer Zero-One Programming Problems." *Discrete Applied Mathematics* 52, 83–106.

Skorin-Kapov, J. (1990). "Tabu Search Applied to the Quadratic Assignment Problem." *ORSA Journal on Computing* 2, 33–45.

Sondergeld, L. and S. Voß. (1996). "A Star-Shaped Diversification Approach in Tabu Search." In I. H. Osman and J. P. Kelly (eds.) *Meta-Heuristics: Theory and Applications*. Dordrecht, Netherlands: Kluwer Academic Publishers, pp. 489–502.

Steinberg, Leon. (1961). "The Backboard Wiring Problem: A Placement Algorithm." *SIAM Review* 3, 37–50.

Stützle, T. and H.H. Hoos. (1999). "The MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems: Towards Adaptive Tools for Global Optimization." In S. Voss, S. Martello, I.H. Osman, C. Roucairol (eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Dordrecht, Netherlands: Kluwer Academic Publishers, pp. 313–329.

Taillard, E.D. (1991). "Robust Taboo Search for the Quadratic Assignment Problem." *Parallel Computing* 17, 443–455.

Taillard, E.D. (1995). "Comparison of Iterative Searches for the Quadratic Assignment Problem." *Location Science* 3, 87–105.

Taillard, E.D. (1998). "FANT: Fast Ant system." *Technical Report IDSIA-46-98*, Lugano, Switzerland: Dalle Molle Institute for Artificial Intelligence.

Taillard, E.D. (2001). "Comparison of Non-Deterministic Iterative Methods." In *Proceedings of MIC'2001–4th Metaheuristic International Conference*, Porto, Portugal, pp. 272–276.

Taillard, E.D. (2002). "Principes D'implémentation des Métaheuristiques." In M. Pirlot and J. Teghem (eds.), *Métaheuristiques et Outils Nouveaux en Recherche Opérationnelle: Méthodes*. Paris,France: Hermès, pp. 55–77.

Taillard, E.D. et al. (1998). "Programmation à mémoire adaptative." *Calculateurs Parallèles, Réseaux et Systèmes Répartis* 10, 117–140.

Taillard, E.D. et al. (2001). "Adaptive Memory Programming: A Unified View of Meta-Heuristics." *European Journal of Operational Research* 135, 1–16.

Tate, David E. and Alice E. Smith. (1995). "A Genetic Approach to the Quadratic Assignment Problem." *Computers and Operations Research* 22, 73–83.