Pesquisa Operacional (2012) 32(1): 165-195

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope

AN EXPERIMENTAL STUDY OF VARIABLE DEPTH SEARCH ALGORITHMS FOR THE QUADRATIC ASSIGNMENT PROBLEM

Elizabeth Ferreira Gouvêa Goldbarg* and Marco César Goldbarg

Received February 11, 2011 / Accepted October 17, 2011

ABSTRACT. This paper introduces a new variable depth search method for the Quadratic Assignment Problem. The new method considers the cost of edges assignment as the criterion to decide which vertices to exchange during local search moves. It also presents the results of an extensive experimental study that compares the performance of local search and variable depth search algorithms for the Quadratic Assignment Problem. The investigation presented here contributes to a better understanding of the potential of these techniques, which are widely used as intensification tools in more sophisticated heuristic methods, such as evolutionary algorithms. Different algorithms presented in the literature were implemented and compared to the proposed methods. The results of a computational experiment with 161 benchmark instances are reported. Different statistical tests are applied in order to analyze the results provided by the experiments.

Keywords: local search, variable depth search, experimental analysis, quadratic assignment problem.

1 INTRODUCTION

Local search algorithms are popular methods to find approximate solutions to hard optimization problems. At present many metaheuristics are extensions or generalizations of local search procedures, such as Tabu Search, Simulated Annealing, GRASP and Variable Neighborhood Search, among others (Glover & Kochenberger, 2003). Local search is also often embedded in evolutionary algorithms to deal with intensification tasks, where the objective is to combine the power of evolutionary operators in determining interesting regions of the search space with the power of local search procedures to conduct a thorough search in a specified region (Grosan & Abraham, 2007). In this sense, investigations of efficient local search methods contribute to the development of better heuristics to hard optimization problems, since the development of more efficient search strategies will result in more powerful heuristics. In its basic form, the local search algorithm starts from an initial solution and searches among the neighboring solutions

*Corresponding author
Universidade Federal do Rio Grande do Norte, Departamento de Informática e Matemática Aplicada, Campus Universitário, Lagoa Nova, 59078-970 Natal, RN, Brasil. E-mails: beth@dimap.ufrn.br; elizabethgoldbarg@gmail.com

one that improves the starting solution. If such a neighbor is found, the process is re-started with the new solution replacing the current one as the starting solution. The iterations continue until no solution with better quality is found in the neighborhood of the current solution. The best solution found by the algorithm is said to be a local optimum with respect to the neighborhood structure used in the search. Usually, the neighborhood structure depends on the problem under consideration and is a determinant factor for the performance of local search algorithms. Variable Depth Search (VDS) is an important variation of local search. It was firstly presented for graph partitioning (Kernighan & Lin, 1970) and for the Traveling Salesman Problem (Lin & Kernighan, 1973). Lin and Kernighan's heuristic for the Traveling Salesman Problem is considered to be one of the most effective methods to symmetric instances (Johnson & McGeoch, 2002). The main feature of VDS is to perform subsequences of local search moves such that large portions of the space of solutions are explored in reasonable processing times. Usually, the search is based on simple neighborhood structures and a sequence of solutions is obtained with the application of simple local search moves. The number of solutions generated in each sequence varies during the search. In most cases, the effectiveness of variable depth searches is highly dependent on the choices of the exchanged elements at each iteration step. Applications of VDS were proposed for the generalized assignment problem (Yagiura *et al.*, 1998) and for bi-partitioning signal flow graphs (de Kock *et al.*, 1995).

Given the trade-off between processing time and quality of solution, a question arises on whether it is advantageous or not to use VDS instead of simple local search approaches. This is the first point investigated in this paper that focuses on the Quadratic Assignment Problem (QAP) which is a strong NP-hard combinatorial optimization problem (Sahni & Gonzalez, 1976) with extensive practical application and an important test ground for most heuristics. A wide range of algorithms based on metaheuristic strategies have been proposed for this problem, many of them have local search methods as an important component (Loiola *et al.*, 2007). Although a huge number of papers are dedicated to the QAP, as far as the authors' knowledge concerns, no paper addressed the investigation of the benefits of using variable depth search methods rather than simple local search methods. This is the first point examined in this paper that compares strategies based on local search and on VDS for the QAP. Two VDS algorithms were proposed for the investigated problem (Li *et al.*, 1994; Rego *et al.*, 2006) which use the 2-exchange neighborhood structure. Those algorithms are compared to the VLSN (Very Large Scale Neighborhood) (Ahuja *et al.*, 2007) and to an iterated local search algorithm that uses the classical 2-exchange neighborhood and was embedded on an efficient Evolution Strategies algorithm (Stützle, 2006). The results of a computational experiment with 92 benchmark instances show that both VDS approaches outperform the local search methods and therefore are a better option as intensification tools for other heuristics.

The VDS algorithms previously proposed in the literature exchange vertices systematically and do not consider any other information besides the value of the objective function to determine which vertices are exchanged. Since the elements that contribute in the computation of the objective function of the QAP are the costs of the assignment of edges resultant from the assignment

of vertices, a new variable depth search heuristic that aims at taking advantage of the information of the cost of edge assignments to choose which vertices are exchanged during the search is proposed here. A computational experiment with 161 benchmark instances shows that the consideration of the cost of the edge assignments is beneficial for several families of instances.

This paper is organized in other five sections besides this one. An overview of the Quadratic Assignment Problem is presented in Section 2. Section 3 presents the methodology used in the computational experiments reported in subsequent sections. Existing VDS and local search algorithms are compared in Section 4. Section 5 introduces the new VDS algorithm and reports the results of computational experiments. Concluding remarks are presented in Section 6.

2 THE QUADRATIC ASSIGNMENT PROBLEM

The Quadratic Assignment Problem was first introduced by Koopmans & Beckman (1957) in the context of facility-location problems. Given square matrices of order n, $F = (f_{ij})$ and $D = (d_{ij})$, the problem consists in finding a permutation ρ of the set $N = \{1, \ldots, n\}$ that minimizes the cost $c(\rho)$ given in equation 1.

$$c(\rho) = \sum f_{\rho(i)\rho(j)} d_{ij} \, . \tag{1}$$

In the location theory, the elements of matrix F represent flows of materials between facilities and the elements of matrix D represent distances between locations. If matrices F and D are symmetric then the QAP is said to be symmetric, otherwise it is asymmetric. In terms of Graph Theory, the QAP can be thought as an assignment between the vertices of two complete graphs of order n, whose weighted adjacency matrices correspond to F and D. The assignment of vertices yields an assignment of the edges of the correspondent graphs. The cost of the assignment of edge $(\rho(i), \rho(j))$ to edge (i, j) is represented by $f_{\rho(i)\rho(j)} d_{ij}$ in equation (1). The objective is to find an assignment of vertices which minimizes the cost of the assignment of edges given by the sum of the costs of each edge assignment.

QAP applications arise in diverse areas such as electronics, chemistry, economy and archeology, among others (Çela, 1998). Some recent applications comprise room allocation problems (Ciriani *et al.*, 2004), visualization of patterns of gene expression (Inostroza-Ponta *et al.*, 2007), web site design (Saremi *et al.*, 2008), layout of machines and facilities in cellular manufacturing (Ramkumar *et al.*, 2008) and channel coding in communication systems (Wang & Wu, 2010). Furthermore, several NP-hard combinatorial optimization problems, such as the traveling salesman, the bin-packing and the max clique problem, can be modeled as QAPs (Çela, 1998).

Algorithmic strategies for solving the QAP to optimality include Branch-and-Bound (Lawler, 1963; Hahn *et al.*, 2001; Anstreicher *et al.*, 2002; Anstreicher, 2003; Zhang *et al.*, 2010), Dynamic Programming (Christofides & Benavent, 1989; Marzetta & Brüngger, 1999), Cutting Plane methods (Bazaraa & Sherali, 1980; Burkard & Bonniger, 1983; Miranda *et al.*, 2005), combinations of Branch-and-Bound and Cutting Plane (Padberg & Rijal, 1996; Kaibel, 1998; Jünger & Kaibel, 2001; Blanchard *et al.*, 2003; Erdogan & Tansel, 2007). Although signifi-

cant research effort has been made in this direction, the most challenging QAP instances solved nowadays are still rather small, $n \leq 40$. Optimal solutions for instances with size $n = 36$ were first reported by Nyström (1999) who solved Ste36b and Ste36c. Anstreicher *et al.* (2002) implemented a Branch-and-Bound algorithm based on a quadratic programming lower bound (Brixius & Anstreicher, 2001) in a computational grid and solved instances up to 32 locations. Zhang *et al.* (2010) propose constraint reductions in linear integer programming QAP formulations and present a Branch-and-Bound algorithm that, although generate more nodes than previous approaches, is more efficient in terms of processing time. They solve instances up to 32 locations on different computational platforms with time limit 4 hours.

Since, usually, real world problems modeled by the QAP are larger than the ones that exact algorithms are able to solve, a wide variety of heuristic approaches were proposed for handling near optimal solutions for this problem (Loiola *et al.*, 2004, 2007). Heuristic algorithms developed according to a myriad of metaheuristic techniques were proposed for the QAP. A review of such algorithms is presented by Loiola *et al.* (2004, 2007). Some recent works include Tabu Search (James *et al.*, 2009a, 2009b; Fescioglu-Unver & Kokar, 2011), Differential Evolution (Davendra *et al.*, 2009), Iterated Local Search (Ramkumar *et al.*, 2009), Ant Colony (Puris *et al.*, 2010) and Lagrangian Smoothing Algorithm (Xia, 2010). Paul (2010) presents a comparison between Simulated Annealing and Tabu Search algorithms for the QAP.

Most successful heuristics presented for the QAP are based on, extend or embed local search methods with the 2-exchange neighborhood structure (Drezner, 2005; Drezner, 2008; James *et al.*, 2009; Fescioglu-Unver & Kokar, 2011). Therefore, an investigation on the potential of local search procedures and variations is very useful in the context of deciding which method to use to compose more sophisticated heuristics.

3 MATERIALS AND METHODS

The next two sections present computational experiments that consider two sets of test cases: QAPLIB (Burkard *et al.*, 1991) (http://www.opt.math.tu-graz.ac.at/qaplib/inst.html), Tai*xx*e and Dre instances (Drezner *et al.*, 2005) (http://mistic.heig-vd.ch/taillard/problemes.dir/qap.dir/qap.html). The QAPLIB is a benchmark library of QAP instances with different structures. Those instances can be divided in four classes (Taillard, 1995): instances randomly generated from a uniform distribution, instances whose distance matrix is based on the Manhattan distance on a grid and the elements of the flow matrix are randomly generated, not necessarily from a uniform distribution, instances that arise from practical applications and instances that have the same characteristics of real-life problems, that is, they are randomly generated, but not from a uniform distribution and the distance matrix corresponds to the Euclidean distance between n points in the plane. Tai*xx*e and Dre instances are introduced by Drezner *et al.* (2005). Those instances were designed to be difficult to solve by algorithms that depend on transposition neighborhoods.

The VDS algorithms utilized in the computational experiments reported in this paper, including the algorithms presented by Li *et al.* (1994) and Rego *et al.* (2006), were implemented under

an Iterated Local Search (ILS) framework. ILS is an effective way of using repeated runs of local search. This method has been proposed, independently, by several researchers who gave it different names, such as iterated descent, large-step Markov chains, and iterated local search among others (Lourenço *et al.*, 2002). This search strategy leads to a randomized walk in the space of local optima and, in many cases, is more effective than the random re-starts. In iterated local search algorithms a starting solution, ρ_0, is generated and a local search is applied to it. The resulting solution, ρ_1, is disturbed and the local search procedure is applied to ρ_1, resulting in ρ_2. An acceptance criterion is applied to decide which solution will be the input for a new application of the local search. These steps are repeated until a given condition is met.

The platform used to implement the iterated local search algorithms is an Intel Core 2, 2.4 GHz, 2 Gb RAM, running Linux. Twenty five independent executions of each algorithm were performed for each instance. The stopping criterion adopted in the experiments is a pre-specified maximum processing time.

In Section 4, existing VDS like algorithms (Li *et al.*, 1994; Rego *et al.*, 2006) are compared to an effective ILS algorithm that uses the classical 2-exchange neighborhood (Stützle, 2006) and to a multi-start algorithm that implements k-exchange neighborhoods (Ahuja *et al.*, 2007). Except for the latter, all algorithms were implemented by the authors. The results reported in the experiments for the multi-start algorithm were presented by Ahuja *et al.* (2007). The experiments were performed on 92 symmetric and asymmetric QAPLIB test cases, including all families of instances.

The computational experiment concerning the VDS algorithms proposed in this paper were performed on 161 symmetric and asymmetric QAP instances with n ranging from 20 to 150: 92 QAPLIB instances, 60 Tai*xx*e instances and 9 Dre instances.

In general, the mean and the median costs of the solutions found by each algorithm on the QAPLIB instances were close. It did not occur with Tai*xx*e instances. Therefore, we opted to present the mean when dealing to QAPLIB instances and the median to Tai*xx*e and Dre instances. The choice to present the median for the latter classes of instances was made since this statistic presents an upper limit to the minimal values found in at least 50% of the independent executions. The results are presented in terms of percent difference from the best known solution calculated with equation 2, where *val* denotes the value of the mean, median or *best* solution accordingly and best denotes the value of the optimal solution or the best known value reported in the literature when the optimum is not known.

$$gap = \frac{val - best}{best} \times 100\,. \tag{2}$$

Statistical analyses were done with the Kruskal-Wallis' test, the Mann-Whitney U test (Conover, 1999) and the test for proportions comparison proposed by Taillard *et al.* (2008) (two-tailed test). In the latter case the proportions refer to number of successes obtained by each heuristic method considering a pre-specified computational effort. Success, here, means to achieve the lowest value when average or median solutions are compared and to achieve the highest value

when dealing with number of best solutions. Algorithms are compared in pairs. The input data for proportions comparison tests are the number of instances considered in a given experiment and the number of instances where each algorithm is successful in relation to the other.

To illustrate the potential of the proposed algorithms, their results are compared to the ones produced with an exact algorithm proposed recently by Zhang *et al.* (2010). The comparison is performed on 17 sparse instances with n up to 32.

4 LOCAL SEARCH AND VARIATIONS

This section reviews some approaches of local search and variations applied to the QAP. The neighborhood structures used in these algorithms are based on the exchange of elements of a current solution. The exchanges are done systematically, without any information, except for the value of the objective function calculated with equation 1. The performances of these algorithms are compared and conclusions are drawn. This section is divided into two parts. The different heuristics are presented in Subsection 4.1 and the computational experiment is presented in Subsection 4.2.

4.1 Local Search for the QAP

Exchange neighborhoods are very popular for problems that can be represented as permutations of n elements, such as the QAP. The local search procedures used in the most successful heuristics proposed to tackle the investigated problem rely on the 2-exchange neighborhood structure. Given a solution ρ, the neighborhood $\aleph(\rho)$ obtained through the 2-exchange structure, is defined by the set of permutations which can be reached by exchanging two elements of ρ, as defined in expression 3.

$$\aleph(\rho) = \left\{\rho' | \rho'[r] = \rho[s], \rho'[s] = \rho[r], \rho'[i] = \rho[i] \; \forall i, i \neq r, s\right\} \tag{3}$$

There are $n(n-1)/2$ possible combinations of two locations of a permutation with n elements and, given two elements, there is only one way of exchanging them. An advantage of this neighborhood for the QAP is that the difference between the costs of two neighbor solutions can be computed in $O(n)$ (Taillard, 1991). This neighborhood is the most widely used in several heuristic algorithms proposed for the examined problem.

The simplest way to use a local search algorithm is to perform several re-starts of the procedure from different points of the search space. Multi-start applications with the 2-exchange neighborhood are reported to the QAP in a number of papers (Fleurent & Glover, 1999; Misevicius, 1997; Misevicius & Riskus, 1999). Resende *et al.* (1996) present a greedy randomized adaptive search (GRASP) for the QAP. An extension is presented by Oliveira *et al.* (2004) who improve the performance of the GRASP algorithm with the inclusion of a path-relinking procedure. They present experimental results for 92 instances of the QAPLIB (Burkard *et al.*, 1991) with n up to 64. Lourenço *et al.* (2003) observe that as the instance size increases, the probability of random re-starts methods to find solutions with significantly lower costs decreases. The Iterated Local

Search method, ILS, is an alternative to prevent that effect. The method is thought to perform a biased sampling of the search space by performing small perturbations in the solution generated by the local search procedure and using this disturbed solution as the starting solution of the next iteration (Lourenço *et al.*, 2003). An ILS approach with the 2-exchange neighborhood is proposed for the QAP by Stützle (2006). The general ILS framework and the ILS algorithm proposed for the QAP are reviewed in the following, since the ILS algorithm proposed by Stützle (2006) is tested in the computational experiments presented in this paper and also the ILS architecture is used in the implementation of the proposed methods. A general framework of the ILS algorithm is presented in Figure 1.

Algorithm *ILS*	
1.	$\rho_0 \leftarrow$ initial_solution( )
2.	$\rho_0 \leftarrow$ local_search(ρ_0)
3.	repeat
4.	$\rho_1 \leftarrow$ disturb(ρ_0, *history*)
5.	$\rho_2 \leftarrow$ local_search(ρ_1)
6.	$\rho_0 \leftarrow$ acceptance_criterion(ρ_0, ρ_2, *history*)
7.	until (*stop criterion is met*)

Figure 1 – General framework of ILS.

An initial solution is generated and a local search is applied to it in steps 1 and 2, respectively. The ILS algorithm may use or not a set to store the history of the search, called *history* in Figure 1. This set may contain, for example, interesting solutions generated during the search. The history of the search may influence the decisions made in steps 4 and 6. Although history can be considered, in many ILS implementations, the decisions regarding the perturbation of the current solution (step 4) and acceptance criterion (step 6) do not depend on it. The solution generated in the first two steps is disturbed and a local search is applied to the new solution in steps 4 and 5, respectively. In order to re-start the search, an acceptance criterion chooses from which solution the search will be continued. At least, two new decisions, beyond those of the local search method, have to be made in order to use an iterated local search algorithm: the method to disturb solutions and the criteria to accept a solution.

Four ILS algorithmic versions were investigated with the major difference between them regarding the criterion to choose the starting solution of the next iteration, called *acceptance criterion*. Those versions are called: *Better*, *Restart*, *RandomWalk* and *LSMC*. In *Better*, the best solution among ρ_0 and ρ_2 is the input for the next iteration. The same acceptance criterion is used in *Restart*, but a new random solution is returned by the acceptance criterion if an improved solution is not found for a fixed number of iterations. In *RandomWalk*, ρ_0 is always replaced by ρ_2 irrespective of the cost of the latter. Finally, in *LSMC* a simulated annealing based criterion is used. In this version, ρ_2 is accepted with a probability p that depends on the difference between the costs of ρ_0 and ρ_2 and on a temperature. The latter version obtains the overall best solutions

when compared to the other three versions and was implemented in our computational experiments for the comparison with the other algorithms. The implementation followed the directions given in the paper it was presented. The local search procedure uses the first pivoting rule and "*don't look bits*". Stützle (2006) presents experimental results of the iterated local search algorithms for 38 instances with n ranging from 20 to 100. The ILS algorithm is then embedded in an evolutionary algorithm that presents high quality results.

Multi-exchange neighborhoods are natural extensions of the 2-exchange. Ahuja *et al.* (2007) investigate multi-exchange neighborhoods for the QAP. They defined ρ', a k-exchange neighbor of ρ, as the permutation obtained from ρ by a cyclic sequence of k elements. Given a sequence of locations $i_1, \ldots, i_k$, the multi-exchange move corresponds to assign facility $\rho(i_j)$ to location i_{j+1}, for all $j \neq k$ and to assign facility $\rho(i_k)$ to location i_1. The local search algorithm iteratively examines all cyclic sequences of increasing values for k, where the maximum is a specified parameter. The size of the neighborhood grows exponentially with k, thus this neighborhood structure is classified as a Very Large Scale Neighborhood, for which an exhaustive examination of the whole neighborhood is impracticable in computational terms. With the use of an improvement graph, Ahuja *et al.* (2007) show that, in average, they can compute the cost of a k-exchange move in $O(k)$. Due to the high processing times needed to the full enumeration scheme the authors established a maximum value for k equals 4. They test multi-start implementations of the proposed neighborhood structure against a multi-start implementation of the 2-exchange neighborhood in 132 test cases of the QAPLIB, with fixed processing times. They show that their neighborhood obtains consistently better results than the multi-start local search with the 2-exchange neighborhood.

In general, for fixed values, as k grows, the computational effort to deal with k-exchange neighborhoods rises rapidly and, usually, the improvement in the quality of solutions does not compensate a greater effort. An effective alternative for the fixed size neighborhood structures is presented by Kernighan & Lin (1970) and Lin & Kernighan (1973) for the Partitioning Problem and the Traveling Salesman Problem, respectively. The idea is to perform sequences of moves, such that each move is likely to lead to a better solution instead of testing all moves of a given neighborhood. This method is called Variable Depth Search (VDS). The VDS algorithm examines iteratively, for growing values of k, whether exchanging k elements may result in a better neighboring solution. A gain function is computed and if it is likely that the previous solution can be improved, then the algorithm tests if k can be extended to $k+1$. When no more gain can be made, the algorithm performs the exchange of the k elements. The general framework of a VDS algorithm is shown in Figure 2.

Similar to other local search algorithms, an initial solution ρ_0 is generated in step 1. The current best solution is stored in ρ_{best} initially set to ρ_0. The variable *gain* stores the result of the gain function. The variable *bestgain* maintains the best value achieved by the gain function during the iterations of the inner loop. Whenever *bestgain* is updated, variable k is set to the value of variable i, the number of the current iteration. The value of k indicates the length of the best sequence of exchanges the algorithm has to perform after the inner loop is executed. Lists *Lout*

Algorithm *VDS*

1. $\rho_0 \leftarrow$ initial_solution( ); $\rho_{best} \leftarrow \rho_0$
2. repeat
3. $i \leftarrow 1$; $gain \leftarrow best\_gain \leftarrow 0$; $k \leftarrow 0$
4. $Lin \leftarrow Lout \leftarrow \{\ \}$
5. $\rho_1 \leftarrow \rho_0$
6. repeat
7. $x_i \leftarrow$ element_out($\rho_1, Lout$); $Lout \leftarrow Lout \cup \{x_i\}$
8. $y_i \leftarrow$ element_in(ρ_1, Lin); $Lin \leftarrow Lin \cup \{y_i\}$
9. $\rho_2 \leftarrow$ exchange(ρ_1, x_i, y_i)
10. $\Delta \leftarrow c(\rho_1) - c(\rho_2)$
11. $gain \leftarrow gain + \Delta$
12. if (*gain* is better than *bestgain*) $bestgain \leftarrow gain$; $k \leftarrow i$
13. $\rho_1 \leftarrow \rho_2$; $i \leftarrow i + 1$
14. until ((*gain* meets *condition*) and (elements_exist()))
15. $\rho_2 \leftarrow$ exchange_sequence(ρ_0, k)
16. if (ρ_2 is better than ρ_{best}) $\rho_{best} \leftarrow \rho_2$
17. $\rho_0 \leftarrow \rho_2$
18. until (*stop criterion is met*)

Figure 2 – General framework of VDS.

and *Lin* contain elements that were withdrawn and included in the current solution, respectively. The choice of elements x_i (that is withdrawn) and y_i (that is included in the current solution) aims at maximizing the improvement of the current solution within the sequence of moves. Variable ρ_1 stores the neighboring solutions found during the VDS iterations. A test is done in step 14 to find out if, given conditions established in advance, the value in *gain* still indicates that an improvement exists with the current sequence of moves. The inner loop also finishes if no elements exist to be exchanged due to the restrictions imposed by the lists of prohibited elements what is verified in procedure *element_exist*( ). In step 15 a new solution ρ_2 is generated with the replacement of elements $x_1, \ldots, x_k$ by $y_1, \ldots, y_k$ in solution ρ_0. The main loop continues iterating until a given stop criterion is met.

A VDS algorithm for the QAP was presented by Li *et al.* (1994) who investigated the complexity of local search for the QAP with a 2-exchange neighborhood which they proved to be PLS-complete. Their algorithm starts with a permutation generated at random, with uniform probability. For a current permutation ρ_0, a sequence of permutations, $\rho_1, \ldots, \rho_s$, is built, each of them obtained from the previous one with a 2-exchange move with cost lower than the current permutation. They used a cumulative gain function, $G(i)$, for permutation ρ_i, where $G(i) = c(\rho_i) - c(\rho_0)$. At each re-start of the local search, ρ_0 is replaced by the best ρ_i, the one with the best $G(i)$. The stopping condition was the inexistence of a sequence for the current permutation. Another VDS like method for the QAP is presented by Rego *et al.* (2006)

who propose an ejection chain algorithm. Their method builds sequences of exchanges until no promising permutations are likely to exist or until a maximum of n moves. Initially, the best 2-exchange move for each facility $\rho(i)$, $i = 1, \ldots, n$, is identified. Let j be the best location for facility $\rho(i)$, then this facility is assigned to location j. The facility that previously occupied position j is said to be ejected and has to be assigned to other location. The method prohibits elements to be moved twice. Thus the sequence can grow until all n facilities are moved. Each element of the move sequence corresponds to one local search step in a QAP instance of q elements, where q varies from n to 1.

4.2 Experiments Comparing Local Search Algorithms

The algorithms by Li *et al.* (1994) and Rego *et al.* (2006) were implemented under an iterated local search framework and are named IT-LPR and IT-EC, respectively. The solution generated by the algorithm presented by Rego *et al.* (2006) is not guaranteed to be a local optimum for the 2-exchange neighborhood. Therefore, a final 2-exchange local search step was included at the end of the iterated local search iterations. Preliminary experiments showed that this modification improved significantly the performance of this algorithm.

The VDS algorithms are compared to the LSMC (Stützle, 2006) and to the VLSN (Ahuja *et al.*, 2007). For the experiments performed in this paper, we tested two methods to update the "*don't look bits*" in the LSMC. The first method followed the directions given in the reference paper by Stützle (2006) where only the "*don't look bits*" of exchanged elements are reset to 0. In the second method all "*don't look bits*" are reset to 0 after a solution is disturbed. The results of the experimentation made to compare these methods showed that the first method produces worse solutions than the second method for the majority of the tested instances. Thus, for comparison purposes, the best result found for each instance by one of these two versions is reported for LSMC.

The results listed in columns related to the VLSN were reported by Ahuja *et al.* (2007) who used an IBM RS6000 platform (333 MHz). Their stopping criterion was 1 hour for problems with $n \geq 40$ and 2 hours for problems greater than 40. The stop condition for LSMC, IT-LPR and IT-EC is the maximum processing time (in seconds) shown in the last column of Tables 1 and 2. These tables present the results for symmetric and asymmetric QAPLIB instances, respectively. The name of the instance is presented in the first column followed by the best known solution in the second column. Columns *Av* and *%best* present, respectively, the average percent deviation from the best known solution and the percentage of best known solutions obtained in 25 independent executions. The results obtained for instances Esc32x, Sko100x and Bur26x are grouped. In these cases the best known solution is not presented in the correspondent column. The best result obtained for each instance by one of the compared algorithms is bold.

VLSN produces the best average percent deviation on instances Esc32e, Esc32g and Tai100a. The LSMC presents the best results on instances Esc32c, Esc32e, Esc32g and Esc64a. The

Table 1 – Comparing local search methods and variants on symmetric QAP instances.

Instance	BKS	VLSN		LSMC		IT-LPR		IT-EC		T(s)
		Av	%best	Av	%best	Av	%best	Av	%best	
Chr20a	2192	33.83	0.02	14.60	0	5.26	**4**	**4.75**	**4**	3
Chr20b	2298	27.59	0	11.22	0	7.42	0	**5.71**	**4**	3
Chr20c	14142	63.75	0.20	18.14	0	3.03	**52**	**2.77**	**52**	3
Chr22a	6156	10.03	0.02	6.46	0	2.92	0	**1.56**	**8**	4
Chr22b	6194	9.55	0	5.95	0	2.59	0	**1.62**	0	4
Chr25a	3796	44.08	0	18.78	0	13.87	0	**4.40**	**36**	6
Esc32a-h	—	4.95	47.80	9.54	50.29	**0.57**	**87.43**	0.69	86.29	5
Esc64a	116	0.14	95.69	**0**	**100**	**0**	**100**	**0**	**100**	15
Esc128	64	5.61	33.89	0.91	28	1.38	64	**0.88**	**80**	20
Had20	6922	0.84	6.76	0.36	36	**0**	**100**	**0**	**100**	3
Kra30a	88900	5.99	0.2	4.73	0	1.22	16	**1.05**	**32**	10
Kra30b	91420	4.13	0.04	2.64	0	0.52	4	**0.12**	**24**	10
Nug20	2570	3.04	0.67	2.02	4	0.01	96	**0**	**100**	3
Nug21	2438	3.14	0.29	1.9	0	**0.03**	**88**	0.07	72	3
Nug22	3596	2.71	1.37	1.53	4	**0**	**100**	0.01	96	4
Nug24	3488	3.31	0.7	1.82	4	0.07	80	**0.04**	**88**	5
Nug25	3744	2.64	0.42	1.66	0	**0.01**	**88**	0.04	68	6
Nug27	5234	3.27	0.4	2.06	0	0.06	64	**0.04**	**96**	7
Nug28	5266	3.3	0.19	2.20	0	0.34	12	**0.28**	**32**	8
Nug30	6124	3.06	0.03	1.96	0	0.35	12	**0.21**	**28**	10
Rou20	725522	3.34	0.02	2.28	0	**0.16**	**20**	0.31	4	3
Scr20	110030	6.22	0.21	2.67	0	**0.04**	88	0.05	**92**	3
Sko42	15812	2.73	0.01	2.24	0	0.55	4	**0.30**	**8**	30
Sko49	23386	2.44	0	2.18	0	0.7	0	**0.46**	0	45
Sko56	34458	2.39	0	2.13	0	0.8	0	**0.54**	0	71
Sko64	48498	2.2	0	2.17	0	0.76	0	**0.47**	0	103
Sko72	66256	2.21	0	2.02	0	0.71	0	**0.50**	0	152
Sko81	90998	1.91	0	2.14	0	0.67	0	**0.50**	0	219
Sko90	115534	1.89	0	2.18	0	0.72	0	**0.59**	0	300
Sko100a-f	—	1.80	0	2.37	0	0.70	**0**	**0.55**	0	415
Ste36a	9526	9.07	0.01	6.51	0	1.85	0	**0.76**	**4**	18
Ste36b	15852	15.95	0.18	7.70	4	1.4	4	**0.08**	**88**	18
Ste36c	8239110	7.24	0	6.12	0	1.17	0	**0.25**	**12**	18
Tai20a	703482	4.37	0.02	3.38	0	**0.67**	**12**	0.84	0	3
Tai25a	1167256	4.08	0	3.26	0	3.30	0	**1.35**	0	6
Tai30a	1818146	3.84	0	3.02	0	1.75	0	**1.46**	0	10
Tai35a	2422002	3.72	0	3.50	0	2.06	0	**1.63**	0	20
Tai40a	3139370	3.68	0	3.51	0	**2.32**	0	2.35	0	25
Tai50a	4938796	3.71	0	3.97	0	**2.84**	0	2.9	0	50

Table 1 (continuation) – Comparing local search methods and variants on symmetric QAP instances.

Instance	BKS	VLSN		LSMC		IT-LPR		IT-EC		T(s)
		Av	%best	Av	%best	Av	%best	Av	%best	
Tai60a	7205962	3.54	0	3.83	0	**2.87**	0	3.03	0	89
Tai80a	13511780	2.78	0	3.32	0	2.74	0	**2.58**	0	223
Tai100a	21052466	**2.49**	0	3.16	0	2.57	0	2.71	0	1000
Tho30	149936	3.72	0.03	3.34	0	0.43	16	**0.3**	**20**	10
Tho40	240516	3.70	0	2.97	0	1.01	0	**0.52**	0	25
Tho150	8133398	2.12	0	3.64	0	0.79	**0**	**0.68**	0	1000
Wil50	48816	1.37	0	1.07	0	0.20	0	**0.11**	0	120
Wil100	273038	0.95	0	3.12	0	0.35	0	**0.25**	0	1000

Table 2 – Results for asymmetric instances.

Instance	BKS	VLSN		LSMC		IT-LPR		IT-EC		T(s)
		Av	%best	Av	%best	Av	%best	Av	%best	
Bur26a-h	——	0.27	1.99	0.26	0.13	0.08	0	**0.00**	**85.50**	15
Lipa20a	3683	2.52	1.37	1.93	12	**0**	**100**	**0**	**100**	3
Lipa30a	13178	1.83	0.24	1.77	0	1.01	24	**0.56**	**60**	4
Lipa40a	31538	1.36	0.01	1.23	0	1.13	0	**0.80**	**28**	25
Lipa50a	62093	1.17	0	1.14	0	1.02	0	**0.93**	**8**	50
Lipa60a	107218	0.99	0	0.99	0	**0.90**	0	**0.90**	0	89
Lipa70a	169755	0.86	0	0.88	0	**0.80**	0	**0.80**	0	150
Lipa80a	253195	0.75	0	0.77	0	**0.71**	0	0.72	0	225
Lipa90a	360630	0.69	0	0.73	0	**0.67**	0	**0.67**	0	300
Lipa20b	27076	12.66	12.94	9.67	32	**0**	**100**	**0**	**100**	3
Lipa30b	151426	14.96	7.34	14.05	12	**0**	**100**	1.80	88	4
Lipa40b	476581	16.9	5.8	17.20	0	**0.66**	**96**	1.39	92	5
Lipa50b	1210244	17.52	2.33	18.90	0	8.26	52	**5.61**	**68**	10
Lipa60b	2520135	19.22	0.42	20.66	0	**16.66**	**12**	17.57	8	17
Lipa70b	4603200	19.94	0.53	21.07	0	**16.65**	**16**	18.32	8	30
Lipa80b	7763962	20.92	0.08	21.74	0	**19.94**	**4**	20.92	0	45
Lipa90b	12490441	—	—	26.61	0	**19.43**	**8**	20.38	4	60
Tai20b	122455319	14.22	6.3	0.79	28	**0**	**100**	**0**	**100**	3
Tai25b	344355646	12.1	0.59	4.66	0	0.07	48	**0**	**100**	15
Tai30b	637117113	9.06	0.13	2.60	0	0.20	4	**0**	**100**	25
Tai35b	283315445	6.47	0.03	4.86	0	0.25	4	**0.09**	**60**	39
Tai40b	637250948	8.29	0.24	6.26	0	0.05	12	**0.00**	**96**	59
Tai50b	458821517	5.73	0	4.15	0	**0.27**	0	**0.27**	**24**	116
Tai60b	608215054	6.16	0	4.80	0	**0.30**	0	0.54	**4**	211
Tai80b	818415043	5.27	0	3.84	0	1.09	0	**0.92**	0	500
Tai100b	1185996137	4.43	0	2.05	0	0.64	0	**0.46**	0	1000
Tai150b	498896643	3.1	0	3.3	0	1.50	0	**1.38**	0	1000

results in Table 1 show that the two VDS algorithms present, in general, the best performance for the 58 symmetric instances of the computational experiment. The IT-LPR presents the best average results on 18 instances. The IT-EC presents the overall best performance regarding quality of solution. It presents 46 best average results including instances Esc32c, Esc32d, Esc32e, Esc32g and the whole Sko100 set. The number of best known solutions produced by the two VDS algorithms is similar with some advantage for the IT-EC. The average concerning the 58 symmetric instances and the 25 independent executions are 4.02, 4.90, 23.65 and 28.39 for the VLSN, LSMC, IT-LPR and IT-EC, respectively.

The Kruskal-Wallis' statistical test was applied to determine if significant differences exist between the results presented by the three ILS algorithms. The VLSN could not be tested, once the results for each instance are not available. The results of the statistical test considering significance level 0.05 showed that the null hypothesis (samples come from identical populations) cannot be rejected only for instances Esc32c, Esc32e, Esc32g, Esc64a and Esc128, indicating that for the other 53 instances the results produced by at least one algorithm were significantly different from the other two. In all cases the test indicated that the LSMC performed significantly worse than the other algorithms. The Mann-Whitney U-test was applied to the results of the two VDS algorithms. With significance level 0.05, the test showed that the IT-EC produced significantly superior results on instances Chr22a, Chr22b, Chr25a, Kra30b, Nug27, Nug28, Tai25a, Tai35a, Sko42, Sko49, Sko56, Sko64, Sko72, Sko81, Sko90, Sko100a-f, Ste36a-c, Tho30, Tho40, Tho150, Wil50 and Wil100. Significant better results were presented by the IT-LPR on instances: Rou20, Tai50a, Tai60a, Tai80a and Tai100a.

In order to apply the test for proportions comparison (Taillard *et al.*, 2008) it is necessary to define "*success*" In this paper, one algorithm is considered successful on one instance when compared to other, if the former produces a better average value than the latter. With significance level 0.01, the statistical test showed that the IT-EC was the best approach when compared to each of the other three algorithms. The VLSN performed conclusively worse than the others. The comparison between the IT-LPR and to the LSMC showed that that the former outperformed the latter.

Similar results were observed on the asymmetric instances presented in Table 2. The best results for these instances were produced by the IT-LPR and the IT-EC. The IT-LPR presented the best average results on 15 instances and the IT-EC presented 26 best average results including Bur26 instances. The average of best known solutions found by each algorithm on the 34 asymmetric instances are 1.55, 3.12, 25.19 and 41.98 for the VLSN, LSMC, IT-LPR and IT-EC, respectively. With significance level 0.05, the Kruskal-Wallis' test showed that the results of, at least, one algorithm is significantly different from the others. The statistical test showed that the two VDS algorithms performed significantly better than the LSMC. Concerning the comparison of the two VDS algorithms, significant best results were obtained by the IT-EC on instances Bur26a-h, Tai25b, Tai30b, Tai35b, Tai40b and Tai100b. IT-LPR presented significant best results on instances Lipa80a, Lipa80b and Lipa90b.

The test for proportions comparison with significance level 0.01 showed that the two VDS algorithms outperform the VLSN and the LSMC. The null hypothesis cannot be rejected (even with significance level 0.05) in the comparison between VLSN and LSMC. The test also pointed out a significant better result of the IT-EC over the IT-LPR.

5 GUIDING VDS WITH THE COSTS OF EDGE ASSIGNMENTS

In terms of Graph Theory, each parcel of equation 1 corresponds to the cost of an assignment of two edges of complete graphs of order n. From this viewpoint, the objective function seeks at minimizing the sum of products of edge weights. The problem of assigning edges is solved in polynomial time, but only few assignments of edges, regarding the total number of such assignments, lead to feasible solutions concerning the assignment of vertices when $n > 3$ (Rangel & Abreu, 2007). Nevertheless, the costs of the edges assignments can be used to guide the search indicating *good* candidates to be exchanged. In this sense, the edges are thought as the exchanging elements, taking due care to maintain feasibility of the solutions produced during the search. Given a solution, ρ, the flow edge e_f with terminal vertices $\rho(i)$ and $\rho(j)$ and the location edge e_d with terminal vertices i and j, the assignment of edges is represented by (e_f, e_d).

Consider that edge e_f is assigned to e_d, the assignment of e_f to a new location e'_d implies in freeing the edge e'_f, previously assigned to e'_d. Therefore, the basic idea consists in, iteratively, canceling an edge assignment and to re-assign the freed flow edge to another distance edge.

In this paper, two alternatives are considered for the new assignment. In the first alternative, the assignment of one terminal vertex of e_f is undone and the assignment of the other terminal vertex remains fixed. In the other alternative both assignments of the terminal vertices of edge e_f are undone. In both cases, the freed vertex(vertices) is(are) assigned to a new location(s). This new assignment induces a new solution, ρ_1. The difference between the costs of the original solution, ρ, and the solution induced by the basic move, ρ_1, is calculated and the gain function is updated. The first and the second alternatives lead to one and two algorithmic versions, respectively. The general framework of the variable-depth search based on the first alternative is presented in algorithm VDS_QAP.

After the initial solution, ρ_0, is generated, the first assignment of edges to be undone is chosen in procedure *edge_assignment*( ). A list of prohibited moves, L_{Prohib}, stores the pair of edges (e_f, e_d) of the assignment that is undone. This list is initialized with (e_f, e_d). Its maximum size is limited to $\#size_{list}$ elements. Inside the loop a new distance edge, e'_d, is chosen. The flow edge e_f is assigned to e'_d in procedure *new_edge*( ). The edge e'_f previously assigned to e'_d is set free. A new solution, ρ_2, is generated by exchanging edges e_f and e'_f. Clearly, the edge assignments depend on the terminal vertices of the corresponding edges. The distinct algorithmic versions proposed in this paper differ in this step, where different strategies to exchange the edges and to assign the terminal vertices are adopted. The variation between the costs of the two solutions is calculated and incremented in variable *gain*. The best value of this variable is stored in variable *bestgain* and the position in the sequence of exchanges is stored in variable k. The list of prohibited assignments is updated with (e'_f, e'_d). If there are less than $\#size_{list}$ elements

Algorithm *VDS_QAP*

$\rho_0 \leftarrow$ *random_solution*()
$(e_f, e_d) \leftarrow$ *edge_assignment*(ρ_0)
$\mathrm{L_{Prohib}} \leftarrow \{(e_f, e_d)\}$; $i \leftarrow 1$; $bestgain \leftarrow 0$
$\rho_1 \leftarrow \rho_0$
repeat
  $e'_d \leftarrow$ *new_edge*(ρ_1, e_f)
  $\rho_2 \leftarrow$ *exchange*(ρ_1, e_f, e'_f)
  $\Delta \leftarrow c(\rho_1) - c(\rho_2)$
  $gain \leftarrow gain + \Delta$
  if ($gain > bestgain$)
    $bestgain \leftarrow gain$; $k \leftarrow i$
  $\mathrm{L_{Prohib}} \leftarrow \mathrm{L_{Prohib}} \cup \{(e'_f, e'_d)\}$
  $e_f \leftarrow e'_f$
  $\rho_1 \leftarrow \rho_2$; $i \leftarrow i + 1$
until ($gain > 0$)
$\rho_1 \leftarrow$ *exchange_sequence*(k, ρ_0)

Figure 3 – General framework of VDS_QAP.

in $\mathrm{L_{Prohib}}$, then the new pair is added to it. Otherwise, the new element replaces the "oldest" element of $\mathrm{L_{Prohib}}$. The main loop iterates while positive gains exist (variable gain is greater than 0). Alternatively a stopping criterion can also consider a maximum number of iterations of the main loop. Finally, the initial solution, ρ_0, is updated with the best sequence of edges exchange in procedure *exchange_sequence*( ).

Two important decisions to be made when implementing VDS_QAP are: to define how many and which assignments are canceled at each iteration step and to define how the new locations for the freed edges are chosen. A number of alternatives exist for these and other implementation issues. Depending on the decisions made by the developer, algorithms with wide varying behaviors are expected to be created. In the next two sections, three algorithmic versions are presented where the details concerning those decisions are explained.

In the algorithms presented in the next sections, the choice of the first edge assignment that is undone is based on the cost of the assignments of each pair of edges in the initial solution. Given a solution, ρ, the weight of the assignment (e_f, e_d) of edge ef with terminal vertices $\rho(i)$ and $\rho(j)$ to edge e_d with terminal vertices i and j is given by $w(e_f, e_d) = f_{\rho(i)\rho(j)} d_{ij}$. In order to select the first pair, (e_f, e_d), a restricted list of candidates is built with the l_{size}% pairs with the greatest weights. One element of that list is randomly chosen with a probability that is directly proportional to the weight of the correspondent assignment.

In order to determine the best value for l_{size}, a preliminary experiment was done with 42 QAPLIB instances with n between 20 and 150. Four values were investigated for l_{size}: 10, 30, 50 and 70.

It was observed that the best results were obtained with 10 or 30, with the latter value yielding slightly better results than the former.

The sequence of movements is terminated if $gain \leq 0$. In the implementations reported here an additional stopping criterion was added to establish a maximum number of iterations being twice the size of L_{Prohib} without the improvement of the best solution found up to the current iteration. The size of L_{Prohib} was set to 150 after preliminary experiments.

5.1 One Terminal Vertex is set Free – VDS1

In the first variant of the proposed algorithm, denoted as VDS1, only one terminal vertex of edge e_f is set free. Thus, it is necessary to decide which terminal vertex assignment remains fixed and which is undone. It corresponds to choose which location of edge e_d becomes unoccupied. This decision is made simultaneously with the choice of the new location for the free facility. Both terminal vertices of edge e_f are tested.

In a first experiment to examine the potential of the variant VDS1 in relation to the 2-exchange neighborhood, 100 random re-starts of each method were applied to all QAPLIB instances with n ranging from 20 to 30. In this experiment, each released vertex was tested in all locations not prohibited for it. The minimal, average and median solution reached by each method were computed, as well as the average processing time. The results are shown in Table 3 where the values of the minimal, average and median solutions are reported in columns *Min*, *Av* and *Med*, respectively. The average processing time in seconds is exhibited in column T.

The results presented in Table 3 show that the minimal values obtained by VDS1 are better than the ones produced by the 2-exchange on 32 of the 38 instances, there are 4 ties and the latter method finds 2 minimal values better than the former. The proposed method also finds the best average solutions on 36 instances. All median values produced by the VDS1 are lower than the median values obtained with the other method. A summary of the improvements in average solutions obtained by VDS1 over the other algorithm per instance class is shown in Table 4. The values exhibited in Table 4 were calculated on classes with at least three instances. Columns *Class*, *Best* and *Improvement* show, respectively, the class name, which algorithm produced the best average value and the improvement of the average solution over the other algorithm.

Table 4 shows that the 2-exchange local search exhibits better results on Tai*xx*b class, where an improvement of 4.01% is obtained. The proposed approach outperforms the 2-exchange algorithm on the remaining instances with the lowest improvement being 16.67% on class Lipa.

Once normality cannot be assumed, the Mann-Whitney test was applied to the results produced by the investigated methods to test whether or not significant differences exist. With significance level 0.01, the null hypothesis cannot be rejected only for four instances: the three Tai*xx*b instances and the Chr20c. Statistical significant differences were found on the remaining instances. Table 3 also shows that smallest processing times are, in general, obtained with the 2-exchange local search. Nevertheless, when the same statistical investigation is applied to the processing times, with significance level 0.05, significant differences exist in favor of the 2-exchange on 10

Table 3 – Results of 100 random starts of methods VDS1 and 2-exchange local search.

Instance	VDS1				2-exchange			
	Min	Av	Med	T(s)	Min	Av	Med	T(s)
Bur26a	0.37	0.87	0.88	0.040	0.60	1.31	1.35	0.010
Bur26b	0.39	0.97	0.94	0.000	0.71	1.43	1.45	0.010
Bur26c	0.49	1.20	1.19	0.050	0.74	1.53	1.47	0.010
Bur26d	0.66	1.50	1.43	0.040	0.97	1.94	2.13	0.020
Bur26e	0.26	1.08	1.06	0.050	0.88	1.77	1.69	0.000
Bur26f	0.25	1.16	1.09	0.030	0.38	1.56	1.47	0.000
Bur26g	0.57	1.37	1.35	0.090	0.59	1.95	1.95	0.000
Bur26h	0.47	1.47	1.35	0.030	0.98	2.15	1.98	0.000
Chr20a	9.40	37.66	37.59	0.040	20.53	48.34	45.76	0.000
Chr20b	8.01	26.50	24.59	0.050	17.84	44.11	42.38	0.000
Chr20c	17.83	74.53	77.99	0.010	16.39	82.24	80.92	0.000
Chr22a	1.88	11.73	12.02	0.020	5.29	13.47	13.09	0.000
Chr22b	2.91	10.16	9.27	0.040	7.85	13.81	14.08	0.000
Chr25a	14.81	47.82	47.10	0.030	27.24	53.65	54.16	0.000
Had20	0	0.94	0.62	0.000	0	1.43	1.33	0.000
Kra30a	1.82	7.03	6.85	0.080	2.14	8.12	8.59	0.000
Kra30b	0.97	4.93	4.71	0.030	2.43	5.61	5.52	0.000
Lipa20a	0	2.47	2.49	0.020	2.25	2.83	2.73	0.000
Lipa20b	0	11.43	14.14	0.030	0	14.97	16.12	0.000
Lipa30a	1.52	1.84	1.83	0.050	1.71	2.00	1.95	0.000
Lipa30b	0	14.62	16.21	0.050	0	15.62	16.72	0.000
Nug20	0.70	2.78	2.80	0.000	0	4.22	4.28	0.000
Nug21	0.16	2.99	2.54	0.030	0.74	5.20	5.08	0.000
Nug22	0	2.39	1.97	0.020	0.95	3.42	3.05	0.000
Nug24	0	3.00	3.04	0.030	0.92	4.13	4.07	0.000
Nug25	0	2.31	2.22	0.070	1.07	3.74	3.50	0.010
Nug27	0	3.25	2.94	0.040	0.04	4.27	3.97	0.000
Nug28	0.19	3.21	3.06	0.040	0.62	3.98	4.14	0.010
Nug30	0.20	2.49	2.27	0.040	0.94	3.76	4.15	0.000
Rou20	0.08	3.05	3.08	0.020	1.64	4.47	4.76	0.000
Scr20	0.03	6.29	5.47	0.060	3.02	11.69	12.61	0.000
Tai20a	0.53	4.24	4.05	0.040	3.64	6.43	6.47	0.000
Tai20b	0	17.82	12.00	0.010	0	15.34	12.29	0.000
Tai25a	1.45	3.91	3.86	0.040	2.81	4.96	4.97	0.000
Tai25b	0.08	15.66	16.46	0.020	0.33	16.32	16.62	0.000
Tai30a	1.57	3.61	3.64	0.050	3.34	5.10	5.04	0.010
Tai30b	0.19	13.50	13.07	0.060	0.93	13.51	13.29	0.010
Tho30	0.53	3.37	3.19	0.090	1.01	4.51	4.63	0.000

Table 4 – Improvements on average solutions per instance class.

Class	Best	Improv
Bur	VDS1	41.79
Chr	VDS1	23.18
Lipa	VDS1	16.67
Nug	VDS1	45.94
Tai*xx*a	VDS1	40.22
Tai*xx*b	2-exchange	4.01

of the 38 instances only. It means that on the majority of instances, 28 of 38, the statistical test did not indicate significant differences regarding computational times.

In the ILS version of VDS1, two matrices are used to establish a preference order among the localities to which a free facility can be assigned. In a pre-processing phase, two matrices $n \times n-1$ of ranks, R_F and R_D, are built from F and D, respectively. In order to generate these matrices, a list is created for each facility(location) that contains the remaining facilities(locations) in non-decreasing(non-increasing) order of flow(distance). Element $R_F[p][q]$ is the position of facility q in the list built for facility p. Element $R_D[p][q]$ is the locality that is in the q-th position of the list correspondent to location p.

Let $\rho(i)$ and $\rho(j)$ be the free and the fixed vertices, respectively, of edge e_f. A list of possible locations for $\rho(i)$, L_{poss}, is built with basis on R_D and element $R_F[\rho(j)][\rho(i)]$. Since j is the occupied location, the first candidate as the new location of facility $\rho(i)$ to be included in L_{poss} is the element in position $R_F[\rho(j)][\rho(i)]$ of the list of locations corresponding to j, that is, element $R_D[j][R_F[\rho(j)][\rho(i)]]$. The other elements of L_{poss} are $R_D[j][R_F[\rho(j)][\rho(i)+1]$, $R_D[j][R_F[\rho(j)][\rho(i)-1], \ldots, R_D[j][R_F[\rho(j)][\rho(i)+k]$, $R_D[j][R_F[\rho(j)][\rho(i)-k]$. If it is the case that $\rho(i)-k=0$ and $\rho(i)+k<n-1$ or $\rho(i)]-k>0$ and $\rho(i)+k=n-1$, L_{poss} is completed with elements $R_D[j][R_F[\rho(j)][\rho(i)+k+1], \ldots, R_D[j][R_F[\rho(j)][n-1]$, or $R_D[j][R_F[\rho(j)][\rho(i)-k], \ldots, R_D[j][R_F[\rho(j)][1]$, respectively. Therefore, L_{poss} is mounted with the locations with best chance of being a good location for $\rho(i)$ according to the values of the corresponding edges. The algorithm seeks the new location for $\rho(i)$, analyzing the exchange of $\rho(i)$ and $\rho(h)$, h being the current location examined of list L_{poss}. In the algorithm described in this paper, a first improvement rule was used. Let bestloc be the best location found for $\rho_(i)$. Then a new solution ρ_1 with cost $c(\rho_1)$ is induced by the interchange of vertices $\rho(i)$ and $\rho(best_{loc})$. The same analysis is done for vertex $\rho(j)$ resulting in solution ρ_2 with cost $c(\rho_2)$. The assignment that induces the solution with the lowest cost among ρ_1 and ρ_2 is assumed as the exchange move.

5.2 Two Terminal Vertexes are Set Free – VDS2

In this second variant, denoted as VDS2, both terminal vertices of edge e_f are re-assigned to other locations. The strategy adopted in VDS1 to define the new location for the free vertex, tends

to be time consuming in this case, once two vertices need to be re-located. Thus, in the VDS2 versions, the choice of the distance edge e'_d to which e_f is assigned has a random element. Two versions of VDS2 are investigated. These variants do not use the list of forbidden movements. This occurs because the probability of undoing previous movements is very small due to the randomness on the choice of the new assignments.

The first version, named VDS2A, in order to define e'_d, one location is selected at random. Let us consider that edge e_d has no facilities assigned to its terminal vertices, since the current facilities will be removed. One facility, fac_1, is chosen at random with equal probability to be assigned to one terminal vertex of e_d. Let i and j be the terminal vertices of edge ed and suppose that fac_1 is assigned to location i. If j is the k-th element in the list of vertex i, regarding the distances between i and the other locations, then the candidate facilities to be assigned to the second free location are close to the k-th element of the corresponding list of fac_1. In the algorithm implemented in this paper, the facilities in positions $k \pm a$ in the list of fac_1 are the candidates to be the facility that will be assigned to location j. In this version, one of these facilities is selected at random with uniform probability. Preliminary tests evaluated values 0, 1 and 2 for variable a. The experiments showed that the results with $a = 0$ were inferior to both $a = 1$ and $a = 2$. A small advantage was observed for $a = 2$ with no significant difference in processing time. Once the two new facilities are chosen, the algorithm determines the best assignment among the two pair of facilities, the ones that left the locations and the new ones.

In the second VDS2 version, VDS2R, e'_d is selected at random among the possible distance edges.

The absence of the best improvement criterion employed in VDS1 reflects on the quality of solutions, but a gain regarding the processing times is obtained. Like in the IT-EC, to guarantee that the solution produced by each algorithm is a local optimum for the 2-exchange neighborhood, a final 2-exchange local search step was included at the end of each ILS iteration.

Tables 5 and 6 present the results of the computational experiments regarding symmetric and asymmetric QAPLIB instances for the iterated local search versions of the three proposed VDS algorithms. The Kruskal-Wallis' statistical test with significance level 0.05 was applied to determine if differences exist between the results presented by the three new algorithms.

The results showed that significant favorable results occur for IT-VDS1 against the other two algorithms on all Bur26 instances. This is also the case for instances Chr20b, Esc128, Sko100a, Tai20a, Tai30a, Tai80a and Tai100b concerning the IT-VDS2A and for instances Lipa80a, Tai40a and Tho150 regarding the IT-VDS2R. The algorithm IT-VDS2A presented significant better results than the algorithm IT-VDS2R on all Bur26 instances and on instances Esc32a, Tai40a and Lipa60a. Better results than the IT-VDS1 were presented by the IT-VDS2A on instances Nug21, Nug24 and Tai100a. Superior results for IT-VDS2R against IT-VDS1 were pointed out on instances Nug24, Sko100d and Lipa30a, and against IT-VDS2A on instances Chr20b, Esc128, Scr20, Sko56, Sko72, Sko100a, Sko100d, Ste36a, Tai80a, Tho40 and Lipa40b.

Table 5 – Edge assignment based VDS for QAPLIB symmetric instances.

Instance	BKS	IT-VDS1		IT-VDS2A		IT-VDS2R		T(s)
		Av	%best	Av	%best	Av	%best	
Chr20a	2192	3.38	**28**	**2.62**	8	3.05	16	3
Chr20b	2298	**5.52**	**8**	6.44	0	5.36	0	3
Chr20c	14142	1.76	68	2.43	**76**	**1.49**	**76**	3
Chr22a	6156	1.29	8	**1.06**	**32**	1.17	20	4
Chr22b	6194	1.72	0	1.82	4	**1.60**	**20**	4
Chr25a	3796	**3.73**	28	5.81	**32**	5.37	12	6
Esc32a-h	—	**0.85**	**83.43**	1.32	82.29	1.16	**83.43**	5
Esc64a	116	**0**	**100**	0.21	92	**0**	**100**	15
Esc128	64	**0.88**	**76**	4.25	32	2.63	60	20
Had20	6922	**0**	**100**	**0**	**100**	**0**	**100**	3
Kra30a	88900	1.18	20	1.28	**28**	**1.10**	24	10
Kra30b	91420	**0.13**	**32**	0.21	24	0.21	24	10
Nug20	2570	**0**	**100**	**0**	**100**	**0**	**100**	3
Nug21	2438	0.09	60	**0.02**	**88**	0.03	84	3
Nug22	3596	**0**	**100**	**0**	**100**	**0**	**100**	4
Nug24	3488	0.13	68	0.04	92	**0.01**	96	5
Nug25	3744	**0.01**	80	0.02	76	0.02	80	6
Nug27	5234	**0**	**100**	**0**	**100**	**0**	**100**	7
Nug28	5166	**0.17**	48	0.26	48	0.22	52	8
Nug30	6124	0.20	20	0.28	8	**0.13**	40	10
Rou20	725522	0.22	4	**0.20**	4	**0.20**	**16**	3
Scr20	110030	0.05	88	0.14	72	**0.01**	**96**	3
Sko42	15812	**0.24**	**20**	**0.37**	4	0.27	4	30
Sko49	23386	0.41	0	**0.37**	0	0.40	0	45
Sko56	34458	0.48	0	0.56	0	**0.42**	0	71
Sko64	48498	0.5	0	**0.47**	0	0.49	0	103
Sko72	66256	**0.49**	0	0.56	0	0.51	0	152
Sko81	90998	0.47	0	0.47	0	**0.44**	0	219
Sko90	115534	0.61	0	0.58	0	**0.55**	0	300
Sko100a-f	—	**0.54**	0	0.58	0	0.55	0	415
Ste36a	9526	0.56	12	0.86	8	**0.52**	**20**	18
Ste36b	15852	**0.01**	**96**	0.11	88	**0.01**	**96**	18
Ste36c	8239110	**0.22**	12	0.31	**32**	0.32	12	18
Tai20a	703482	0.72	2	**0.52**	**3**	0.60	**3**	3
Tai25a	1167256	**1.29**	1	1.30	0	1.47	0	6
Tai30a	1818146	**1.52**	0	1.79	0	1.55	0	10
Tai35a	2422002	1.76	0	1.90	0	**1.73**	0	20
Tai40a	3139370	**2.22**	0	2.45	0	2.73	0	25
Tai50a	4938796	**2.85**	0	2.97	0	3.06	0	50

Table 5 (continuation) – Edge assignment based VDS for QAPLIB symmetric instances.

Instance	BKS	IT-VDS1		IT-VDS2A		IT-VDS2R		T(s)
		Av	%best	Av	%best	Av	%best	
Tai60a	7205962	**2.95**	0	3.05	0	3.10	0	89
Tai80a	13511780	**2.48**	0	2.95	0	2.89	0	223
Tai100a	21052466	2.77	0	**2.69**	0	2.73	0	1000
Tho30	149936	**0.21**	**40**	0.24	28	0.27	32	10
Tho40	240516	0.65	0	0.62	0	**0.45**	0	25
Tho150	8133398	**0.66**	0	0.77	0	0.75	0	1000
Wil50	48816	0.12	**4**	0.11	0	**0.09**	0	120
Wil100	273038	0.26	0	**0.25**	0	**0.25**	0	1000

Table 6 – Edge assignment based VDS for QAPLIB asymmetric instances.

Instance	BKS	IT-VDS1		IT-VDS2A		IT-VDS2R		T(s)
		Av	%best	Av	%best	Av	%best	
Bur26a-h	—	**0.01**	**95.5**	0.05	0.13	0.30	0	15
Lipa20a	3683	**0**	**100**	**0**	**100**	**0**	**100**	3
Lipa30a	13178	0.79	44	0.54	64	**0.24**	**84**	4
Lipa40a	31538	0.83	28	0.94	16	**0.73**	**36**	25
Lipa50a	62093	1.03	0	**0.9**	**12**	0.92	8	50
Lipa60a	107218	0.91	0	**0.89**	0	0.92	0	89
Lipa70a	169755	0.81	0	0.81	0	0.80	0	150
Lipa80a	253195	**0.72**	0	0.73	0	0.73	0	225
Lipa90a	360630	0.68	0	0.67	0	0.67	0	300
Lipa20b	27076	**0**	**100**	**0**	**100**	**0**	**100**	3
Lipa30b	151426	0.59	96	0.62	96	**0**	**100**	4
Lipa40b	476581	2.04	88	3.60	76	**0**	**100**	5
Lipa50b	1210244	7.71	56	8.82	48	**5.64**	**68**	10
Lipa60b	2520135	16.84	12	16.11	16	**15.32**	**20**	17
Lipa70b	4603200	**15.94**	**20**	15.96	20	16.75	16	30
Lipa80b	7763962	20.06	4	20.08	4	**19.27**	**8**	45
Lipa90b	12490441	21.23	0	21.31	0	**18.73**	**12**	60
Tai20b	122455319	**0**	**100**	**0**	**100**	**0**	**100**	3
Tai25b	344355646	**0**	**100**	**0**	**100**	**0**	**100**	15
Tai30b	637117113	**0**	**100**	**0**	**100**	**0**	**100**	25
Tai35b	283315445	**0.07**	**72**	0.09	52	0.10	56	39
Tai40b	637250948	0.24	88	0.15	84	**0**	**100**	59
Tai50b	458821517	0.22	**36**	0.23	16	**0.21**	28	116
Tai60b	608215054	0.72	0	0.72	4	**0.48**	**4**	211
Tai80b	818415043	0.75	0	0.79	0	**0.71**	0	500
Tai100b	1185996137	**0.29**	0	0.54	0	0.35	0	1000
Tai150b	498896643	**1.42**	0	1.43	0	1.52	0	1000

The test for proportions comparison with significance level 0.05 shows that, except for the pair IT-VDS1 and IT-VDS2R, significant differences do exist among the other pairs of algorithms. Considering the averages presented in Tables 1 and 5, that test also shows that the IT-LPR presents significant worse results than the three proposed VDS algorithms in the symmetric QAPLIB instances. IT-VDS2A is outperformed by IT-VDS1, IT-VDS2R and IT-EC. The algorithms ITVDS1 and ITVDS2R present better results than the IT-EC. The algorithm IT-VDS1 shows a particular good performance on instances Tai*xx*a where the algorithm presents the best results on six of the nine instances. The algorithms IT-VDS2R and IT-VDS2A present 22 and 13 best average results and 13 and 11 best percentages of the best known solution, respectively. Considering the 25 independent executions of each algorithm for each instance of the experiment, the algorithm ITVDS2R obtains, in average, 31.12 % of the best known solutions, followed by IT-VDS1, 29.92% and IT-VDS2A, 28.90%. These three results are better than the ones presented by algorithms IT-LPR and IT-EC.

Considering the average values presented in Table 6, the test for proportions comparison with significance level 0.05 shows that, the algorithm IT-VDS2R presents significantly better results than the other VDS algorithms on the set of asymmetric instances. The test also points out the best performance of IT-VDS2R over IT-LPR and IT-EC and that the latter outperforms the IT-VDS1. A significant difference is also found in favor of IT-VDS1 against IT-VDS2A. With the specified significance level, it is not possible to point out differences between the other results.

The results concerning the number of best known solutions of the asymmetric instances obtained by the proposed algorithms show that, in average, the algorithm ITVDS2R obtains the best performance with 42.22% of the best known solutions, followed by IT-VDS1 with 42.20%, IT-EC with 41.98%, IT-VDS2A with 38.77% and IT-LPR with 25.19%.

5.3 Comparison of the proposed VDS heuristics with an exact algorithm

To illustrate the difference between the results obtained with exact algorithms and those obtained with the methods proposed in this paper, a comparison is presented in Table 7. The comparison is made with an exact algorithm recently proposed by Zhang *et al.* (2010) to solve sparse QAP instances. Columns *Min* present the percent difference from the optimal solution and the best solution found by the corresponding algorithm. Column $T(s)$ of the exact algorithm shows the processing time to obtain the optimal solution or the maximum processing time given as stop criterion for the algorithm. An asterisk means that the exact algorithm reached the maximum processing time without solving the corresponding instance. The exact algorithm was executed by its authors on a laptop Intel Pentium M-1.70GHz processor, 1.23Gb RAM. Columns $T(s)$ of the VDS algorithms show the average processing time to find the optimal solution. An asterisk in those columns means that the corresponding heuristic algorithm reached the maximum processing time without finding the optimal solution. Values 0.00 in columns $T(s)$ mean that the average processing time was less than 0.01s. Columns %*best* show the percentage of best known solutions obtained in the 25 independent executions.

Table 7 – Comparison with the results of an exact algorithm.

Instance	Exact Algorithm		IT-VDS1			IT-VDS2A			IT-VDS2R		
	Min	T(s)	Min	%best	T(s)	Min	%best	T(s)	Min	%best	T(s)
Chr20a	0	28.27	0	28	0.88	0	8	1.04	0	16	1.04
Chr20b	0	56.00	0	8	1.00	2.35	0	*3.00	2.35	0	*3.00
Chr20c	0	313.60	0	68	0.36	0	76	0.68	0	76	0.33
Chr22a	0	21.40	0	8	2.16	0	32	1.96	0	20	1.83
Chr22b	0	15.10	0	4	1.80	0	4	1.88	0	20	2.13
Chr25a	0	274.00	0	28	3.32	0	32	3.13	0	12	4.00
Esc32a	49.23	*14400.00	1.54	0	*5.00	1.54	0	*5.00	1.54	0	*5.00
Esc32b	78.57	*14400.00	0	100	0.84	0	76	2.16	0	84	1.84
Esc32c	11.53	*14400.00	0	100	0.00	0	100	0.00	0	100	0.00
Esc32d	18.00	*14400.00	0	100	0.00	0	100	0.64	0	100	0.72
Esc32e	0	5.90	0	100	0.00	0	100	0.00	0	100	0.00
Esc32g	0	100.20	0	100	0.00	0	100	0.00	0	100	0.00
Esc32h	0	14400.00	0	100	0.00	0	100	0.84	0	100	0.56
Scr20	0.59	*14400.00	0	88	0.40	0	72	0.52	0	96	0.83
Kra30a	20.75	*14400.00	0	20	2.56	0	28	2.72	0	24	4.04
Kra30b	23.23	*14400.00	0	32	4.40	0	24	4.32	0	24	6.00
Kra32	34.71	*14400.00	0	44	4.96	0	44	6.40	0	28	5.04

The two ITVDS2 algorithms did not find the optimal solution of the Chr20b instance and no algorithm found the optimal solution of Esc32a instance. The results produced by the exact algorithm on Chr instances are very effective, as shown in Table 7. However on the remaining instances, the heuristic algorithms are always preferable to the exact algorithm. Even on the Esc32e instance, where the exact algorithm takes only 5.9s to solve it, the heuristic algorithms take, in average, less than 0.01s to find the optimal solution and find the optimum in all executions. The exact algorithm cannot find the optimal solution of eight instances. From those instances, the heuristic algorithm misses the optimum only on instance Esc32a, yet the difference to the optimal value is much smaller as well as the processing time. Another point to consider is that the heuristic algorithms find optimal solutions on instances larger than those considered by the exact algorithms currently proposed for the QAP.

5.4 Testing with Dre*xx* instances

Instances with n between 15 and 72 were tested. Table 8 shows averages and percentage of optimal solutions obtained by each algorithm. The maximum processing times in seconds are presented in column $T(s)$.

As shown in Table 8, IT-LPR behaves poorly on this set of instances presenting results significantly inferior to the other algorithms. The Kruskal-Wallis' test performed on the results of the other four algorithms show that, with significance level 0.05, no significant differences were found between their performances on instances Dre from 15 to 30. Significant differences in

Table 8 – Results for Drexx instances.

Id	BKS	IT-EC		IT-LPR		IT-VDS1		IT-VDS2A		IT-VDS2R		T(s)
		Av	% best	Av	% best	Av	% best	Av	% best	Av	% best	
Dre15	306	0	100	381.83	0	0	100	0	100	0	100	15
Dre18	332	0	100	414.65	0	0	100	0	100	0	100	18
Dre21	356	8.18	68	428.54	0	**4.36**	**80**	5.39	76	4.97	76	21
Dre24	396	23.23	16	443.76	0	21.41	16	**20.60**	**24**	24.67	4	24
Dre28	476	26.74	12	450.72	0	30.24	0	**26.32**	**16**	29.03	8	28
Dre30	508	40.39	0	448.71	0	**38.25**	0	44.83	0	45.98	0	30
Dre42	764	**62.39**	0	471.19	0	67.33	0	67.52	0	69.46	0	42
Dre56	1086	**82.42**	0	470.85	0	89.22	0	88.53	0	90.12	0	56
Dre72	1452	**92.70**	0	474.42	0	98.37	0	99.11	0	99.24	0	152

favor of IT-EC were found on instances Dre42, 56 and 72 in comparison to the other three algorithms. The test for proportions comparison confirms the best performance of IT-EC on Dre instances in comparison to the IT-VDS2R. Evidences of significant differences are not pointed out by the proportions comparison test in pairwise comparisons of the other algorithms.

5.5 Testing with Tai*xx*e instances

Instances with $n = 27$, 45 and 75 are tested. The maximum processing times given to the algorithms are 10 seconds ($n = 27$), 30 seconds ($n = 45$) and 200 seconds ($n = 75$). Tables 9, 10 and 11 show the results for classes $n = 27$, 45 and 75, respectively. The first column contains the identification of each test case. The second column shows the optimum or the best known solution of each instance. Columns *Md* show the percent deviation of the median from the optimum or best known solution. Columns %*best* show the percentage of optimal or best known solutions found by each algorithm.

With significance level 0.05, the Kruskal-Wallis' test pointed out significant differences on the performance of the algorithms for all instances of class 27, where the inferior performance of IT-LPR was identified. Due to this fact, other comparisons were carried on the results obtained only with the other four algorithms. The success for this set of instances is established on the number of best solutions found by each algorithm. With significance level 0.05, the test for proportion comparison showed that IT-EC and IT-VDS1 are outperformed by IT-VDS2R. In general, IT-VDS1 is outperformed by IT-EC and IT-VDS2A, although the hypothesis test does not point out significant differences. IT-VDS2A and IT-EC performs similarly.

Significant differences between the results presented by the IT-LPR and the other algorithms in class Tai45e instances were also pointed out by the Kruskal-Wallis' test. The Mann-Whitney test with significance level 0.05 indicated that better results were found by algorithm IT-VDS2A against IT-EC on instances 7 and 11, by algorithm IT-EC against IT-VDS2A on instance 9, by IT-VDS1 against IT-EC on instance 9, and by algorithm IT-VDS2R against IT-VDS2A on instances 6 and 14. In average, the best performance regarding the number of solutions with cost

Table 9 – Results for Tai27e instances.

Id	BKS	IT-EC		IT-LPR		IT-VDS1		IT-VDS2A		IT-VDS2R	
		Md	%best	Md	%best	Md	%best	Md	%best	Md	%best
1	2558	0	**96**	1.17	32	0	84	0	84	0	84
2	2850	0	**88**	9.02	0	0	72	0	**88**	0	76
3	3258	0	**100**	4.79	4	0	88	0	**100**	0	**100**
4	2822	0	**100**	3.54	16	0	**100**	0	**100**	0	**100**
5	3074	0	84	7.61	4	0	92	0	84	0	**100**
6	2814	0	**100**	3.62	48	0	96	0	92	0	88
7	3428	0	84	1.98	32	0	**88**	0	**88**	0	84
8	2430	0	**100**	5.69	20	0	96	0	**100**	0	**100**
9	2902	0	80	5.31	12	0	68	0	76	0	**88**
10	2994	0	**100**	7.82	12	0	96	0	92	0	**100**
11	2906	0	**80**	1.38	48	0	72	0	72	0	**80**
12	3070	0	80	9.45	8	0	88	0	**92**	0	84
13	2966	0	68	0.27	44	0	76	0	80	0	**88**
14	3568	0	88	4.34	16	0	96	0	88	0	**96**
15	2628	0	76	9.09	8	0	**80**	0	**80**	0	**80**
16	3124	0	**100**	0	60	0	**100**	0	96	0	**100**
17	3840	0	80	1.95	20	0	**96**	0	**96**	0	92
18	2758	0	76	3.63	28	0	68	0	**96**	0	68
19	2514	0	80	7.92	0	0	60	0	76	0	**88**
20	2638	0	60	3.94	24	0	68	0	68	0	**80**

equals to BKS is obtained by the IT-VDS1, followed by the IT-VDS2A, IT-EC and IT-VDS2R as shown in Table 10. The test for proportions comparison indicated that IT-VDS1 performs better than IT-VDS2R in the Tai45e instances with confidence level 93.38%. The test did not detect significant differences among the performances of the remaining algorithms.

The Kruskal-Wallis' test pointed out significant differences between the results presented by at least one algorithm on class Tai75e instances. The Mann-Whitney U-test showed that the IT-LPR performed worse than the IT-VDS1 and IT-VDS2R on 13 instances, the IT-EC on 14 instances and the IT-VDS2A on 17 instances. The test for proportions comparison considering the medians also showed significant differences in favor of IT-VDS1, IT-VDS2A, IT-VDS2R and IT-EC against IT-LPR. At significance level 0.05, the test for proportions comparison showed that IT-EC, IT-VDS2R and IT-VDS2A are outperformed by IT-VDS1. The test does not indicate significant differences among the other three algorithms.

In general, it was observed that the relative performance of IT-VDS1 improves for growing sizes of Tai*xx*e instances, being better than IT-LPR on all sets and better than the IT-EC on the set $n = 75$. Moreover, the relative performance of the IT-VDS2R worsens as the sizes of the instances grow. This fact shows that the systematic search produces better results on these classes for increasing sizes of instances. All VDS algorithms performed better than the IT-LPR.

Table 10 – Results for Tai45e instances.

Id	BKS	IT-EC		IT-LPR		IT-VDS1		IT-VDS2A		IT-VDS2R	
		Md	%best	Md	%best	Md	%best	Md	%best	Md	%best
1	6412	7.29	44	10.01	0	**0**	**52**	**0**	**52**	**0**	**52**
2	5734	**0**	**76**	14.82	**0**	0	64	**0**	68	**0**	60
3	7438	**0.40**	**48**	15.86	0	8.71	32	1.64	36	8.71	36
4	6698	**0**	**52**	12.57	0	7.61	44	6.99	32	7.61	28
5	7274	8.33	28	15.04	0	8.33	40	**0**	32	8.33	**52**
6	6612	1.75	48	20.75	0	7.56	40	7.56	**52**	**0**	28
7	7526	3.96	40	7.28	4	**0**	60	**0**	60	**0**	**72**
8	6554	**0**	**60**	13.7	0	**0**	56	5.98	32	5.98	44
9	6648	4.51	36	15.4	0	**0**	**68**	**0**	44	4.51	56
10	8286	**0**	**76**	8.35	8	**0**	60	1.71	60	**0**	44
11	6510	5.87	32	14.1	0	**0**	52	**0**	56	**0**	**64**
12	7510	**0**	**64**	13.36	8	0.27	48	0	52	**0**	52
13	6120	7.91	**48**	11.99	4	8.50	44	4.34	44	**1.41**	40
14	6854	**0**	72	18.76	4	**0**	72	0.09	**84**	**0**	48
15	7394	**4.00**	36	14.66	0	**4.00**	**44**	**4.00**	40	**4.00**	40
16	6520	6.9	44	12.02	4	**0**	56	**0**	52	**0**	**60**
17	8806	6.45	36	13.83	0	**0**	56	4.36	**64**	**0**	40
18	6906	**0**	56	15.32	8	0	56	0	48	0.32	**64**
19	7170	**0**	52	8.01	4	**0**	**64**	**0**	60	**0**	60
20	6510	5.1	40	16.84	0	**0**	**60**	5.1	36	5.1	24

6 CONCLUDING REMARKS

This paper presented an extensive experimental investigation that compared local search and variable depth search algorithms for the Quadratic Assignment Problem. Moreover, variable depth search algorithms that use information concerning the cost of the edges assignments were presented. This study contributes to a better understanding of the potential of these techniques, which are widely used as intensification tools in more sophisticated heuristic methods, such as evolutionary algorithms. The experiments were performed with 161 benchmark QAP instances belonging to classes designed with different structures. Nonparametric statistical tests were applied to the data generated by the computational experiments to support conclusion about the performance of the algorithms.

Existing VDS-like algorithms for the QAP (Li *et al.*, 1994; Rego *et al.*, 2006) were implemented under an iterated local search framework and submitted to a computational experiment more extensive than the ones reported in their original papers. These algorithms were compared with recent approaches that use the classical 2-exchange neighborhood (Stützle, 2006) and k-exchange neighborhoods Ahuja *et al.* (2007). They were also compared to the VDS algorithms proposed in this paper.

Table 11 – Results for Tai75e instances.

Id	BKS	IT-EC		IT-LPR		IT-VDS1		IT-VDS2A		IT-VDS2R	
		Md	%best	Md	%best	Md	%best	Md	%best	Md	%best
1	14488	274.76	0	24.67	0	274.23	0	**12.00**	0	19.16	**4**
2	14444	285.11	0	291.86	0	27.25	0	**12.73**	0	15.58	0
3	14154	265.11	0	274.4	0	18.68	**8**	265.21	4	**13.34**	0
4	13694	280.2	**4**	287.72	0	280.04	0	62.39	0	**19.18**	**4**
5	12884	16.80	0	27.86	0	16.38	**8**	**15.34**	0	274.39	0
6	12554	24.91	0	31.29	0	**15.62**	**4**	27.33	0	21.48	0
7	13782	310.17	0	317.28	0	**9.11**	0	11.8	0	10.85	0
8	13948	286.81	0	292.63	0	**12.46**	0	284.01	0	13.64	0
9	12650	**7.18**	**4**	24.52	0	17.14	**4**	18.78	0	17.74	0
10	14192	**11.95**	0	289.18	0	281.47	0	14.40	0	16.69	0
11	15250	17.15	0	21.73	0	282.20	0	**11.99**	**8**	17.65	0
12	12760	**14.12**	0	296.08	0	15.09	0	16.24	0	16.69	0
13	13024	268.20	**4**	24.55	0	17.06	**4**	**14.00**	0	268.49	0
14	12604	**16.41**	**4**	290.81	0	19.95	0	279.21	0	279.21	0
15	14294	**13.67**	0	51.14	0	14.57	0	17.76	0	16.09	**4**
16	14204	281.32	0	24.25	0	282.13	**4**	**10.46**	**4**	283.62	0
17	13210	21.89	**8**	23.56	0	266.19	**8**	270.84	0	**15.28**	4
18	13500	289.59	0	23.69	0	289.66	0	**16.25**	0	18.53	0
19	12060	15.39	0	27.23	0	**13.17**	**4**	22.32	0	24.28	**4**
20	15260	24.67	0	26.54	0	**14.34**	**4**	16.22	**4**	46.16	**4**

The experiments showed that, given the same computational effort, variable depth search is always preferable to pure local search approaches. The test showed that both previously proposed variable depth search methods outperformed LSMC (Stützle, 2006) and VLSN (Ahuja *et al.*, 2007). Among these four methods, the best performance was obtained by the iterated local search version of the algorithm presented by Rego *et al.* (2006).

The paper introduced VDS like algorithms that consider the cost of edge assignments as a criterion to decide which elements are exchanged while searching the space of solutions of QAP instances. The experiments showed that the proposed criterion was beneficial for several classes of benchmark instances, indicating that the approaches introduced here are preferable to the existing ones on those classes. Statistical tests showed that the proposed algorithms are always preferable to the approach presented by Li *et al.* (1994). Those tests also confirmed that the IT-VDS1 version is better than an ILS version of the algorithm presented by Rego *et al.* (2006) on QAPLIB classes Ste, Chr, Tai*xx*a, Nug and Tai75*e* being outperformed by the latter on Lipa*xx*a class.

Future works will investigate statistical indicators in the choice of edges to be exchanged in variable depth search algorithms.

ACKNOWLEDGEMENTS

We thank anonymous reviewers for their suggestions and useful remarks which contributed to improve the paper. We also acknowledge the support of CNPq to this research under grants 303538/200-2 and 300778/2010-4.

REFERENCES

[1] AHUJA RK, JHA KC, ORLIN JB & SHARMA D. 2007. Very large-scale neighborhood search for the quadratic assignment problem. *Informs Journal on Computing*, **19**(4): 646–657.

[2] ANSTREICHER KM, BRIXIUS NW, GOUX J-P & LINDEROTH J. 2002. Solving large quadratic assignment problems on computational grids. *Mathematical Programming*, **91**(3): 563–588.

[3] ANSTREICHER KM. 2003. Recent advances in the solution of quadratic assignment problems. *Mathematical Programming Ser. B*, **97**: 27–42.

[4] BAZARAA MS & SHERALI HD. 1980. Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem. *Naval Research Logistics Quarterly*, **27**: 29–41.

[5] BLANCHARD A, ELLOUMI S, FAYE A & WICKER N. 2003. Un algorithme de génération de coupes pour le problème de l'affectation quadratique. *INFOR: Information Systems and Operational Research*, **41**(1): 35–49.

[6] BURKARD RE & BONNIGER T. 1983. A heuristic for quadratic Boolean programs with applications to quadratic assignment problems. *European Journal of Operation Research*, **13**: 374–386.

[7] BURKARD R, KARISCH S & RENDL F. 1991. QAPLIB – A quadratic assignment problem library. *European Journal of Operational Research*, **55**: 115–119. Online version on http://www.opt.math.tu-graz.ac.at/qaplib.

[8] BRIXIUS NW & ANSTREICHER KM. 2001. Solving quadratic assignment problems using convex quadratic programming relaxations. *Optimization Methods and Software*, **16**(1-4): 49–68.

[9] ÇELA E. 1998. *The Quadratic Assignment Problem.* Kluwer Academic Publishers.

[10] CIRIANI V, PISANTI N & BERNASCONI A. 2004. Room allocation: A polynomial subcase of the quadratic assignment problem. *Discrete Applied Mathematics*, **144**: 263–269.

[11] CONOVER WJ. 1999. *Practical Nonparametric Statistics.* Third Ed., Wiley.

[12] CHRISTOFIDES N & BENAVENT E. 1989. An exact algorithm for the quadratic assignment problem on a tree. *Operations Research*, **37**(5): 760–768.

[13] DAVENDRA D, ZELINKA I & ONWUBOLU G. 2009. Clustered population differential evolution approach to quadratic assignment problem, in: *IEEE CEC 2009 Congress on Evolutionary Computation*, **1**: 1224–1231.

[14] DREZNER Z. 2005. The extended concentric tabu for the quadratic assignment problem. *European Journal of Operational Research*, **160**(2): 416–422.

[15] DREZNER Z. 2008. Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. *Computers & Operations Research*, **35**(3): 717–736.

[16] DREZNER Z, HAHN P & TAILLARD E. 2005. Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. *Annals of Operations Research*, **139**(1): 65–94.

[17] ERDOGAN G & TANSEL B. 2007. A branch-and-cut algorithm for quadratic assignment problems based on linearizations. *Computers and Operations Research*, **34**(4): 1085–106.

[18] FESCIOGLU-UNVER N & KOKAR MM. 2011. Self controlling tabu search algorithm for the quadratic assignment problem. *Computers & Industrial Engineering*, **60**: 310–319.

[19] FLEURENT C & GLOVER F. 1999. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, **11**: 189–203.

[20] GLOVER F & KOCHENBERGER G. 2003. Handbook of Metaheuristics, Springer.

[21] GROSAN C & ABRAHAM A. 2007. Hybrid evolutionary algorithms: Methodologies, architectures, and reviews. Hybrid Evolutionary Algorithms. *Studies in Computational Intelligence*, **75**: 1–17.

[22] HAHN PM, HIGHTOWER WL, JOHNSON TA, GUIGNARD-SPIELBERG M & ROUCAIROL C. 2001. Tree elaboration strategies in branch and bound algorithms for solving the quadratic assignment problem. *Yugoslav Journal of Operations Research*, **11**: 41–60.

[23] INOSTROZA-PONTA M, MENDES A, BERRETTA R & MOSCATO P. 2007. An integrated QAP-based approach to visualize patterns of gene expression similarity. *Lecture Notes in Computer Science*, **4828**: 156–167.

[24] JAMES T, REGO C & GLOVER F. 2009a. A cooperative parallel tabu search algorithm for the quadratic assignment problem. *European Journal of Operational Research*, **19**: 810–826.

[25] JAMES T, REGO C & GLOVER F. 2009b. Multistart tabu search and diversification strategies for the quadratic assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, **39**(3): 579–596.

[26] JOHNSON DS & MCGEOCH LA. 2002. Experimental analysis of heuristics for the STSP, in: GUTTIN G & PUNNEN A. (Eds.), *The Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, Dordrecht, 369–443.

[27] JÜNGER M & KAIBEL V. 2001. Box-inequalities for quadratic assignment polytopes. *Mathematical Programming*, **91**(1): 175–197.

[28] KAIBEL V. 1998. Polyhedral combinatorics of quadratic assignment problems with less objects than locations. *Lecture Notes in Computer Science*, **1412**: 409–422.

[29] KERNIGHAN BW & LIN S. 1970. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, **49**: 291–307.

[30] DE KOCK EA, AARTS EHL, ESSINK G, JANSEN REJ & KORST JHM. 1995. A variable-depth search for the recursive bipartitioning of signal flow graphs. *OR Spectrum*, **17**(2-3): 159–172.

[31] KOOPMANS TC & BECKMANN MJ. 1957. Assignment problems and the location of economic activities. *Econometrica*, **25**: 53–76.

[32] LAWLER EL. 1963. The quadratic assignment problem. *Management Science*, **9**: 586–599.

[33] LI Y, PARDALOS PM & RESENDE MGC. 1994. A greedy randomized adaptive search procedure for the quadratic assignment problem, in: PARDALOS PM & WOLKOWICZ H. (Eds.), Quadratic

Assignment and Related Problems. *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, **16**: 237–261.

[34] LIN S & KERNIGHAN BW. 1973. An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research*, **21**: 498–516.

[35] LOIOLA EM, ABREU NMM, BOAVENTURA-NETTO PO, HAHN P & QUERIDO TM. 2007. A survey for the quadratic assignment problem. *European Journal of Operational Research*, **176**(2): 657–690.

[36] LOIOLA EM, ABREU NMM & BOAVENTURA-NETTO PO. 2004. Uma revisão comentada das abordagens do problema quadrático de alocação. *Pesquisa Operacional*, **24**(1): 73–110.

[37] LOURENÇO HR, MARTIN OC & STÜTZLE T. 2003. Iterated local search, in: GLOVER F & KOCHENBERGER G. (Eds.), Handbook of Metaheuristics. *International Series in Operations Research & Management Science*, **57**: 321–353. Springer.

[38] MARZETTA A & BRÜNGGER A. 1999. A dynamic-programming bound for the quadratic assignment problem. *Lecture Notes in Computer Science*, **1627**: 339–348.

[39] MIRANDA G, LUNA HPL, MATEUS GR & FERREIRA RPM. 2005. A performance guarantee heuristic for electronic components placement problems including thermal effects. *Computers & Operations Research*, **32**: 2937–2957.

[40] MISEVICIUS A. 1997. A new algorithm for the quadratic assignment problem. *Information Technology and Control*, **5**: 39–44.

[41] MISEVICIUS A & RISKUS A. 1999. Multistart threshold accepting: Experiments with the quadratic assignment problem. *Information Technology and Control*, **12**: 31–39.

[42] NYSTRÖM M. 1999. *Solving Certain Large Instances of the Quadratic Assignment Problem: Steinberg's Examples*. Caltech Computer Science Technical Report caltechCSTR:2001.010, Department of Computer Science, California Institute of Technology Pasadena CA.

[43] OLIVEIRA CA, PARDALOS PM & RESENDE MGC. 2004. GRASP with path-relinking for the quadratic assignment problem, in: RIBEIRO CC & MARTINS SL. (Eds.), Efficient and Experimental Algorithms. *Lecture Notes in Computer Science*, **3059**: 356–368.

[44] PADBERG MW & RIJAL MP. 1996. *Location, Scheduling, Design and Integer Programming.* Kluwer.

[45] PAUL G. 2010. Comparative performance of tabu search and simulated annealing heuristics for the quadratic assignment problem. *Operations Research Letters*, **38**: 577–581.

[46] PURIS A, BELLO R & HERRERA F. 2010. Analysis of the efficacy of a two-stage methodology for ant colony optimization: Case of study with TSP and QAP. *Expert Systems with Applications*, **37**: 5443–5453.

[47] RAMKUMAR AS, PONNAMBALAM SG, JAWAHAR N & SURESH RK. 2008. Iterated fast local search algorithm for solving quadratic assignment problems. *Robotics and Computer-Integrated Manufacturing*, **24**: 392–401.

[48] RAMKUMAR AS, PONNAMBALAMB SG & JAWAHAR N. 2009. A new iterated fast local search heuristic for solving QAP formulation in facility layout design. *Robotics and Computer-Integrated Manufacturing*, **25**: 620–629.

[49] RANGEL MC & ABREU NMM. 2007. Free poset on permutations by scalar products. *Congressus Numerantium*, **184**: 173–184.

[50] REGO C, JAMES T & GLOVER F. 2006. *An Ejection Chain Algorithm for the Quadratic Assignment Problem.* School of Business Administration, University of Mississippi, MS.

[51] RESENDE MGC, PARDALOS PM & LI Y. 1996. Algorithm 754: FORTRAN subroutines for approximate solution of dense quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software*, **22**: 104–118.

[52] SAHNI S & GONZALEZ T. 1976. P-complete approximation problems. *Journal of the Association of Computing Machinery*, **23**: 555–565.

[53] SAREMI HQ, ABEDIN B & KERMANI AM. 2008. Website structure improvement: Quadratic assignment problem approach and ant colony meta-heuristic technique. *Applied Mathematics and Computation*, **195**: 285–298.

[54] STÜTZLE T. 2006. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, **174**: 1519–1539.

[55] TAILLARD ED. 1991. Robust tabu search for the quadratic assignment problem. *Parallel Computing*, **17**: 443–455.

[56] TAILLARD ED. 1995. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, **3**: 87–105.

[57] TAILLARD E, WAELTI P & ZUBER J. 2008. Few statistical tests for proportions comparison. *European Journal of Operational Research*, **185**: 1336–1350.

[58] WANG X & WU X. 2010. Index assignment optimization for joint source-channel MAP decoding. *IEEE Transactions on Communications*, **58**(3): 901–910.

[59] XIA Y. 2010. An efficient continuation method for quadratic assignment problems. *Computers & Operations Research*, **37**: 1027–1032.

[60] YAGIURA M, YAMAGUCHI T & IBARAKI T. 1998. A variable depth search algorithm with branching search for the generalized assignment problem. *Optimization Methods and Software*, **10**(2): 419–441.

[61] ZHANG H, BELTRAN-ROYO C & CONSTANTINO M. 2010. Effective formulation reductions for the quadratic assignment problem. *Computers & Operations Research*, **37**: 2007–2016.