A Variable Depth Search Algorithm for the Quadratic Assignment Problem

Elizabeth F. G. Goldbarg and Marco C. Goldbarg

Universidade Federal do Rio Grande do Norte, Natal, RN, Brazil,
Departamento de Informática e Matemática Aplicada
beth@dimap.ufrn.br, gold@dimap.ufrn.br

Abstract. The Quadratic Assignment Problem arises in many real world applications and is known to be NP-hard. Several heuristics have been proposed for handling near optimum solutions to its instances. Most of those heuristics rely on neighborhood based search strategies, the most popular of them being those which exchange k elements of a solution. However, a drawback for those approaches is that, for growing values of k, the computational effort to explore such neighborhoods rises rapidly. Variable depth neighborhoods can be interesting alternatives for these problems. In this paper, a stochastic variable depth search strategy is presented. The proposal is implemented under an iterated local search framework and compared with related approaches presented previously in the literature. Computational experiments show that the proposed algorithm presents consistently better performance than their counterparts, showing that the proposed strategy is an attractive alternative to be embedded in algorithms based on metaheuristics.

1 Introduction

The Quadratic Assignment Problem (QAP) has been introduced by Koopmans and Beckman [7] 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 $N = \{1, ..., n\}$ that minimizes the cost $c(\rho)$ given in equation 1.

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

In the location theory, the elements f_{ij} of matrix F represent the flow of materials between facilities i and j and the elements d_{ij} of matrix D represent the distance from location i to location j. 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 undirected complete graphs of order n, whose weighted adjacency matrices correspond to F and D. The assignment of vertices yields an assignment of edges of the correspondent graphs. The cost of the assignment of edge e_f to

edge e_d with weights $w(e_f)$ and $w(e_d)$, respectively, is given by $w(e_f) \times w(e_d)$. 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.

The importance of QAP is due to three main factors: its practical application, several NP-hard combinatorial optimization problems can be modeled as QAPs, and it is a strong NP-hard problem. A review of the QAP is presented by Loiola et al. [10]. Several heuristic algorithms were proposed for this problem. The most successful ones are based on metaheuristic techniques. Many of those approaches are local search extensions [3] [8] [13], or hybrid techniques that embed local search procedures [4].

In local search algorithms, exchange neighborhoods are very popular for problems that can be represented as permutations of n elements, such as the QAP. The 2-exchange neighborhood structure is used in the most successful heuristics proposed for the investigated problem. Given a solution ρ, its neighborhood, $\aleph(\rho)$, according to the 2-exchange structure, is defined by the set of permutations which can be obtained by exchanging two elements of ρ. Extensions of the 2-exchange neighborhood, k-exchange neighborhoods, are investigated by Ahuja *et al.* [1] for the QAP. In their paper, they develop a very large-scale neighborhood (VLSN) structure, presenting algorithms that examine all exchanges of nodes for increasing values of k, where the maximum value is a parameter. A generic search procedure to enumerate and evaluate neighbors is developed and several specific implementations of the generic procedure are proposed. They compare their proposal with the 2-exchange neighborhood on benchmark symmetric and asymmetric QAP instances.

In general, as the number k of exchanging elements grows, the computational effort 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, called *variable-depth neighborhood*, was presented by Lin and Kernighan [9]. The implementation of a variable depth search algorithm depends on a number of decisions. Depending on the strategies adopted by programmers concerning those decisions, the distinct implementations may result on algorithms with wide varying behaviors. Applications of variable-depth search were proposed for the generalized assignment problem [14] and for the bi-partitioning of signal flow graphs [6]. A variable-depth neighborhood for the QAP is sketched in the paper of Glover and Rego [5]. Their method builds sequences of facilities exchanges until no promising permutations are likely to exist or until a maximum of n moves.

The main feature of the variable-depth search 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. A sequence of solutions is obtained with the application of simple local search neighborhoods of size s, where s 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. Usually, the variable-depth searches use deterministic criteria to make decisions at each iter-

ation step, choosing elements to be exchanged that allows the best improvement of the current solution. In general, the solutions produced by those algorithms are local optima regarding the basic neighborhood structure they use. Although this strategy to make decisions guarantees that output solutions are local optima, the algorithm may get stuck on these solutions.

In this paper, we propose a variable-depth neighborhood that includes random decisions concerning the elements to be exchanged. Although in the basic form of the proposed search algorithm, the output solution is not guaranteed to be a local optimum, this strategy allows sampling different regions of the space of solutions and prevents the algorithm to get stuck on local optima. Another feature of the algorithm proposed in this paper is that, unlike other algorithms that are based solely on the exchange of vertices, the neighborhood used for the QAP is based also on the exchange of edges.

Section 2 describes the variable depth search strategy and presents the proposed approach. Section 3 gives a brief introduction of iterated local search algorithms, presents an ILS algorithm proposed previously for the QAP [12], once the results of that algorithm are compared with the ones obtained by the algorithm presented here, and presents the ILS version of the algorithm proposed here. Computational experiments are reported in section 4, where a comparison with the results of a multi-start algorithm that implements a generalization of the 2-exchange neighborhood, named Very Large Scale Neighborhood, proposed by Ahuja et al. [1], is also presented. Finally, some conclusions are drawn in section 5.

2 Variable-depth local search

Instead of testing all moves of neighborhood structures where a fixed value k of elements are exchanged, the variable-depth search, VDS, performs sequences of moves, such that each move is likely to lead to a better solution. At each iteration step of a VDS, the algorithm examines, for growing values of k, whether exchanging k elements may result in a better neighbor 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 $k + 1$. In the VDS basic form, if it appears that no more gain can be made, the algorithm performs the exchange of the k elements. The general framework of the variable-depth search as proposed by Lin and Kernighan [9] is presented in algorithm LK. The algorithm has a main loop where an initial solution is generated to be the starting solution of the VDS implemented in the inner loop.

In principle, each neighbor chosen at each VDS iteration, is a minimum cost neighbor of the previous solution. Once this strategy favors cycling, lists of prohibited moves are maintained such that previous movements are not undone. In algorithm LK those lists are denoted by L_{out} and L_{in}. These lists store, respectively, the elements that leave the solution and the elements that are added to it. Similar to other local search algorithms, an initial solution, ρ_0, is generated at random. The current best solution is stored in ρ_{best} that, initially, is set to

ρ_0. The variable *gain* stores the result of the gain function. The variable $best_{sol}$ maintains the best value achieved by the gain function during the iterations of the inner loop. Whenever $best_{sol}$ is updated, variable k receives 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. The choice of the element x_i (that is withdrawn) and the element y_i (that is added to the current solution), aims at maximizing the improvement of the current solution when the sequence of k moves, where elements $x_1, ..., x_k$ are replaced by $y_1, ..., y_k$, is performed. Variable ρ_1 stores the neighbor solutions found during the VDS iterations. The inner loop stops if certain conditions established *a priori* for the gain function are met. The inner loop also finishes if no elements exist to be exchanged due to the restrictions imposed by the lists of prohibited elements.

Pseudo-code of LK

```
ρ0 ← random_solution(); ρbest ← ρ0
repeat
  i ← 1; gain ← bestsol ← 0; k ← 0
  Lin ← Lout ← {}
  ρ1 ← ρ0
  repeat
    xi ← element_out(ρ1); Lout ← Lout ∪ {xi}
    yi ← element_in(ρ1) ; Lin ← Lin ∪ {yi}
    ρ2 ← exchange(ρ1, xi, yi)
    Δ ← c(ρ1) − c(ρ2)
    gain ← gain + Δ
    if (gain is better than bestsol)
      bestsol ← gain; k ← i
    ρ1 ← ρ2; i ← i + 1
  until(gain meets condition)
  ρ1 ← exchange_sequence(k, ρ0)
  if (ρ1 is better than ρbest)
    ρbest ← ρ1
  ρ0 ← update_initial_solution()
until(stopping_criterion is met)
```

Besides usual decisions that have to be made for ordinary local search algorithms, in implementing a variable-depth search one has far more choices to make. Some of those choices concern: the criterion to choose x_i and y_i, the gain function, the criterion that points out whether an improvement is likely to occur or not, the partial move to be extended (the one that produces the best value of the objective function or the one with the best value of the gain function) and the existence of a bound on the search depth, among others.

Each parcel of equation 1 corresponds to the cost of an assignment of two edges of two complete undirected graphs of order n. Once the assignment of

vertices always leads to an assignment of edges, the algorithms proposed to tackle the QAP presented in the literature, usually focus on vertices as the main elements to guide the search. In this paper, exchanges of edges are used to guide vertices exchanges. This is due to the fact that the edges are the main elements for the cost computation. Once an assignment of edges may not lead to an assignment of vertices, it is necessary to guarantee that feasible solutions are generated in order to conduct a search in the space of solutions.

Given a solution, ρ, let edge e_f of the flow graph with terminal vertices $\rho(i)$ and $\rho(j)$ be assigned to edge e_d of the distance graph with terminal vertices i and j. The assignment of edges is represented by (e_f, e_d). The assignment of e_f to a new edge 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 re-assign the freed *flow edge* to other *distance edge*. The list of prohibited moves stores the pair of edges of the undone assignment (e_f, e_d).

There are several possibilities for the new assignment and, in this paper, one of these possibilities is examined. In this alternative, the assignment of the terminal vertices of edge e_f are undone. The freed edge e_f has to be assigned to a new edge e'_d. The latter edge is selected randomly among those edges that are not prohibited to be assigned to e_f. The algorithm checks the possible assignments of vertices between the two edges and chooses the one which induces the best solution.

Pseudo-code of VDS_QAP

```
ρ0 ← random_solution();
(ef,ed)←edge_assignment(ρ0)
L_Prohib ← {(ef, ed)}; i ← 1
ρ1 ← ρ0
repeat
   e'd ←new_edge(ρ1, ef)
   ρ2 ← exchange(ρ1, ef, e'f)
   Δ ← c(ρ1) − c(ρ2)
   gain ← gain + Δ
   if (gain is better than best_sol)
      best_sol ← gain; k ← i
   L_Prohib ←update_list(L_Prohib, (e'f, e'd))
   ef ← e'f
   ρ1 ← ρ2; i ← i + 1
until (condition is met)
ρ1 ← exchange_sequence(k, ρ0)
```

The general framework of a variable-depth search based on edge assignments for the QAP 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*(). The list of prohibited moves, L_{Prohib}, is initialized with (e_f, e_d). In this work, a limited size list with $size_{list}$ elements is used. Inside

the loop a new distance edge, e'_d, is chosen to be assigned to e_f in procedure *new_edge*(). Once edge e'_d is selected, a new edge e'_f that is set free is determined. Clearly, the edge assignments depend on the terminal vertices of the correspondent edges. The basic movement of exchanging edges e_f and e'_f in the current solution is done, giving rise to a new solution, ρ_2. The list of prohibited assignments is updated with (e'_f, e'_d). In procedure *update_list*(), if there are less elements than $size_{list}$ in L_{Prohib}, then the new element is added to this list. Otherwise, the new element replaces the "oldest" element of L_{Prohib} that is implemented as a circular queue.

3 Iterated Local Search

Iterated local search, ILS, is an effective way of using repeated local search iterations with the output of the previous iteration being used to re-start the search [11]. In ILS algorithms, an initial solution ρ_0 is generated and a local search is applied to it. Then, the local optima obtained from ρ_0 is disturbed generating solution ρ_1. The latter solution is submitted to a local search resulting on solution ρ_2. The process is re-started from solution ρ_0 or ρ_2, depending on an acceptance criterion. The algorithm iterates until a stopping criterion is satisfied. In addition to the decisions that have to be made in local search algorithms, at least, two new decisions have to be made in an ILS algorithm: the method to disturb solutions and the criteria to choose from which solution the process is re-started (acceptance criterion).

Four versions of an iterated local search based on the 2-exchange neighborhood are presented by Stützle [12] for the QAP. The major difference between them regards the acceptance criterion. The version that obtains the overall best solutions when compared to the other three versions is called *LSMC*. It is used in our computational experiments for the comparison with the proposed approach. In *LSMC* a simulated annealing based acceptance criterion is used. A solution ρ_2 is accepted with a probability p that depends on a *temperature* and on the difference between the costs of the initial solution ρ_0 and ρ_2. The implementation of *LSMC* follows the directions given in [12] and are described in the following. Procedure *local_search*() uses the first pivoting rule and "don't look bits". The perturbation exchanges k elements chosen at random. The input parameters of procedure *disturb*() are the current solution and k, the number of elements that will be exchanged. During the iterations, the value of k varies between k_{min} and k_{max} in a variable neighborhood search fashion. That is, if after the perturbation and the subsequent local search no better solution is found, k is increased by one. The values of k_{min} and k_{max} are, respectively, 3 and $\max\{0.9n,50\}$. In procedure *acceptance_criterion*(), if $c(\rho_2) < c(\rho_0)$, then ρ_2 is accepted with probability p equals 1. Otherwise, the value of p is given by $p = e^{\frac{c(\rho_0)-c(\rho_2)}{T}}$, where T is the temperature. The temperature begins with value T_{init} and is lowered during the iterations. The value of T_{init} is set to $0.025c(\rho_0)$ after ρ_0 is locally optimized. The temperature is lowered every 10 iterations according to a geometric cooling scheme where $T_{i+1} = 0.9T_i$.

In the iterated local search version of the VDS proposed here, called *It-VDS*, the VDS_QAP procedure replaces the local search step of the ILS algorithm. The perturbation procedure exchanges k random elements of the current solution, however, unlike the $LSMC$, k has a fixed value. Distinct values were investigated for k in preliminary experiments and the value of $0.5n$ was adopted. In the acceptance criterion procedure of the proposed algorithm, the current solution is always replaced by the solution generated in the variable-depth search step. Once there is no guarantee that the local optima generated by the proposed neighborhood are also local optima regarding the 2-exchange neighborhood, a 2-exchange local search step is included in the main loop of this ILS algorithm.

4 Computational Experiments

The platform used to implement the iterated local search algorithms is a Pentium IV, 3.0 GHz, 1 Gb RAM, running Linux. The results of the *It-VDS* are compared with the *LSMC*, and with the results published by Ahuja et al. [1] concerning the very large scale neighborhood algorithms, named *VLSN1* and *VLSN2*.

Two versions of the *LSMC* were implemented in this work to check whether the "don't look bits" update in the procedure that disturbs the current solution improve the results obtained by the algorithm. The first version follows the reference work and only the "don't look bits" of exchanged elements are reset to 0. In the second version all "don't look bits" are reset to 0 after the solution is disturbed. The best result obtained with one of these two versions is reported for each instance. The results of *VLSN1* and *VLSN2* were reported by Ahuja et al. [1] who used an IBM RS6000 platform (333 MHz). The stopping criterion they used was 1 hour for problem sizes $n \leq 40$ and 2 hours for problem sizes greater than 40.

The results for 33 asymmetric instances, table 1, and for 48 symmetric instances, table 2, refer to 25 independent executions of each iterated local search algorithm and each instance. The columns of these tables show the name of the instance, the best solution used to compare the algorithms, BKS, the average percent deviation from the solution reported in column BKS and the percentage of best solutions found by the algorithms. Columns T_{av} show the average processing time in seconds of the LSMC and It-VDS algorithms and column T_{max} shows the maximum processing time in seconds given for each iterated local search algorithm. Comparing the iterated local search approaches where the 2-exchange (*LSMC*), and the proposed random variable depth neighborhood (*It-VDS*) were used, the tables show that except for instance Esc128, the proposed algorithm was able to find the best average results of all instances. The non-parametric statistic test, named U-test [2], was applied to verify the hypothesis of equality of the average solutions presented by the LSMC and the It-VDS. With a level of significance of 0.01, the test showed that the null hypothesis could not be rejected only for instances Esc128, Lipa40a-90a, and Lipa90b.

The algorithm VLSN1 obtains one average result better than the It-VDS for instance Bur26d. For the remaining 80 instances, the proposed approach obtains

Table 1. Results for asymmetric instances

Instance	BKS	*VLSN1*		*VLSN2*		*LSMC*			*It-VDS*			$T_{max}(s)$
		Av	%best	Av	%best	Av	%best	$T_{av}(s)$	Av	%best	$T_{av}(s)$	
bur26a	5426670	0.24	**1.11**	0.32	0.32	2.18	0.00	4.04	**0.08**	0.00	9.92	15
bur26b	3817852	0.30	**0.58**	0.41	0.29	2.32	0.00	5.88	**0.03**	0.00	9.38	15
bur26c	5426795	0.26	**1.05**	0.41	0.31	2.53	0.00	4.96	**0.06**	0.00	6.83	15
bur26d	3821225	**0.29**	**0.54**	0.47	0.28	3.35	0.00	5.77	0.30	0.00	9.58	15
bur26e	5386879	0.20	**2.65**	0.37	0.39	2.31	0.00	5.50	**0.04**	0.00	8.13	15
bur26f	3782044	0.24	**3.07**	0.46	0.66	3.18	0.00	5.50	**0.03**	0.00	9.50	15
bur26g	10117172	0.25	**2.44**	0.36	0.47	2.70	0.00	7.42	**0.06**	0.00	8.29	15
bur26h	7098658	0.35	**4.50**	0.46	0.95	3.33	0.00	5.96	**0.04**	0.00	9.67	15
lipa20a	3683	2.52	1.37	2.80	0.50	1.93	12.00	0.38	**0.82**	**64.00**	0.38	3
lipa30a	13178	1.83	0.24	2.02	0.03	1.77	0.00	1.79	**0.25**	**88.00**	1.08	4
lipa40a	31538	1.36	0.01	1.51	0.00	1.23	0.00	17.92	**0.71**	**36.00**	15.67	25
lipa50a	62093	1.17	0.00	1.30	0.00	1.14	0.00	34.25	**0.99**	**4.00**	41.38	50
lipa60a	107218	0.99	0.00	1.10	0.00	0.99	0.00	70.08	**0.92**	0.00	61.42	89
lipa70a	169755	0.86	0.00	0.95	0.00	0.88	0.00	130.79	**0.80**	0.00	108.29	150
lipa80a	253195	0.75	0.00	0.83	0.00	0.77	0.00	193.50	**0.73**	0.00	153.04	225
lipa90a	360630	0.69	0.00	0.77	0.00	0.73	0.00	262.13	**0.67**	0.00	205.44	300
lipa20b	27076	12.66	12.94	14.80	4.96	9.67	32.00	0.38	**3.26**	**84.00**	0.29	3
lipa30b	151426	14.96	7.34	15.94	4.03	14.05	12.00	1.96	**0.00**	**100.00**	0.25	4
lipa40b	476581	16.90	5.80	18.23	2.03	17.20	4.00	3.58	**0.00**	**100.00**	1.92	5
lipa50b	1210244	17.52	2.33	18.40	0.70	18.90	0.00	8.17	**5.87**	**68.00**	6.54	10
lipa60b	2520135	19.22	0.42	19.71	0.22	20.66	0.00	14.83	**15.16**	**20.00**	11.54	17
lipa70b	4603200	19.94	0.53	20.37	0.17	21.07	0.00	26.67	**15.77**	**16.00**	19.29	30
lipa80b	7763962	20.92	0.08	21.23	0.06	21.74	0.00	39.13	**19.19**	**12.00**	33.12	45
tai20b	122455319	14.22	6.30	15.13	0.88	0.79	28.00	0.38	**0.00**	**100.00**	0.17	3
tai25b	344355646	12.10	0.59	15.53	0.08	4.66	0.00	3.92	**0.00**	**100.00**	1.21	15
tai30b	637117113	9.06	0.13	12.65	0.00	2.60	0.00	11.04	**0.00**	**100.00**	6.83	25
tai35b	283315445	6.47	0.03	8.12	0.01	4.86	0.00	20.63	**0.08**	**60.00**	10.92	39
tai40b	637250948	8.29	0.24	10.18	0.01	6.26	0.00	32.83	**0.00**	**100.00**	18.92	59
tai50b	458821517	5.73	0.00	7.10	0.00	4.15	0.00	91.58	**0.21**	**28.00**	89.21	116
tai60b	608215054	6.16	0.00	7.75	0.00	4.80	0.00	177.71	**0.48**	**4.00**	160.88	211
tai80b	818415043	5.27	0.00	6.01	0.00	3.84	0.00	457.77	**0.71**	0.00	440.67	500
tai100b	1185996137	4.43	0.00	5.20	0.00	2.05	0.00	570.08	**0.35**	0.00	783.04	1000
tai150b	498896643	3.10	0.00	3.44	0.00	3.30	0.00	992.00	**1.52**	0.00	900.44	1000

better average values than the VLSN versions. From the 33 asymmetric instances the two VLSN versions outperform the It-VDS regarding the best values for percentage of best solutions found for the 8 instances of class Bur. The It-VDS obtains the best percentages of 43 instances. If a factor of 10 is used to multiply the processing times of algorithm It-VDS in order to have an approximation between the processing times of the two platforms, then it is possible to observe by the values reported in tables 1 and 2, that the processing times produced by the proposed algorithm are smaller than the VLSN versions, except for instances Tai100a, Tai100b and Tai150b.

5 Conclusion

This paper presented a new proposal in variable depth search for the QAP where both edges and vertices are considered to build the neighborhood of the local search algorithm. The results were compared to results published previously for the QAP, concerning related approaches.

Table 2. Results for symmetric instances

Instance	BKS	*VLSN1*		*VLSN2*		*LSMC*			*It-VDS*			$T_{max}(s)$
		Av	%best	Av	%best	Av	%best	$T_{av}(s)$	Av	%best	$T_{av}(s)$	
Chr20a	2192	33.83	0.02	43.11	0.00	14.48	0.00	0.92	**2.83**	**16.00**	1.04	3
Chr20b	2298	27.59	0.00	33.18	0.00	11.22	0.00	1.25	**5.36**	0.00	1.33	3
Chr20c	14142	63.75	0.20	68.17	0.23	18.14	0.00	0.67	**1.49**	**76.00**	0.33	3
Chr22a	6156	10.03	0.02	12.73	0.00	6.46	0.00	0.96	**1.17**	**20.00**	1.83	4
Chr22b	6194	9.55	0.00	11.86	0.00	5.95	0.00	1.58	**1.60**	**20.00**	2.13	4
Chr25a	3796	44.08	0.00	52.94	0.00	19.01	0.00	3.42	**4.55**	**28.00**	3.45	6
Esc128	64	5.61	33.89	13.09	8.26	**0.91**	28.00	3.38	**1.82**	**60.00**	12.08	20
Had20	6922	0.84	6.76	1.17	2.29	0.36	36.00	0.29	**0**	**100.00**	0.08	3
Kra30a	88900	5.99	0.20	7.70	0.02	4.73	0.00	3.50	**0.60**	**56.00**	4.46	10
Kra30b	91420	4.13	0.04	5.76	0.00	2.64	0.00	3.75	**0.15**	**56.00**	4.33	10
Nug20	2570	3.04	0.67	3.97	0.26	2.02	4.00	0.42	**0**	**100.00**	0.33	3
Nug21	2438	3.14	0.29	4.35	0.14	1.90	0.00	0.38	**0.02**	**80.00**	0.67	3
Nug22	3596	2.71	1.37	3.59	0.53	1.53	4.00	0.67	**0**	**100.00**	0.25	4
Nug24	3488	3.31	0.70	4.45	0.26	1.81	4.00	1.21	**0.01**	**96.00**	0.71	5
Nug25	3744	2.64	0.42	3.80	0.06	1.66	0.00	2.20	**0.02**	**80.00**	1.54	6
Nug27	5234	3.27	0.40	4.16	0.11	2.06	0.00	1.92	**0**	**100.00**	1.63	7
Nug28	5166	3.30	0.19	4.32	0.02	2.20	0.00	2.08	**0.23**	**52.00**	2.83	8
Nug30	6124	3.06	0.03	4.10	0.01	1.96	0.00	4.70	**0.13**	**40.00**	6.42	10
Rou20	725522	3.34	0.02	4.36	0.01	2.28	0.00	0.50	**0.20**	**16.00**	0.75	3
Scr20	110030	6.22	0.21	6.52	0.18	2.67	0.00	0.83	**0.01**	**96.00**	0.83	3
Sko42	15812	2.73	0.01	3.48	0.00	2.24	0.00	23.54	**0.27**	**4.00**	25.04	30
Sko49	23386	2.44	0.00	3.03	0.00	2.18	0.00	36.54	**0.40**	0.00	35.00	45
Sko56	34458	2.39	0.00	2.92	0.00	2.13	0.00	63.67	**0.42**	0.00	55.67	71
Sko64	48498	2.20	0.00	2.68	0.00	2.17	0.00	94.04	**0.49**	0.00	87.58	103
Sko72	66256	2.21	0.00	2.64	0.00	2.02	0.00	140.75	**0.51**	0.00	122.71	152
Sko81	90998	1.91	0.00	2.26	0.00	2.14	0.00	203.63	**0.44**	0.00	176.00	219
Sko90	115534	1.89	0.00	2.22	0.00	2.18	0.00	283.12	**0.55**	0.00	239.63	300
Sko100a	152002	1.74	0.00	2.07	0.00	1.89	0.00	389.33	**0.46**	0.00	332.50	415
Sko100b	153890	1.71	0.00	2.02	0.00	2.34	0.00	399.88	**0.56**	0.00	292.33	415
Sko100c	147862	1.96	0.00	2.29	0.00	2.61	0.00	403.46	**0.46**	0.00	312.67	415
Sko100d	149576	1.74	0.00	2.07	0.00	2.41	0.00	398.71	**0.64**	0.00	331.67	415
Sko100e	149150	1.94	0.00	2.30	0.00	2.65	0.00	393.42	**0.57**	0.00	389.67	415
Sko100f	149036	1.71	0.00	2.02	0.00	2.32	0.00	393.96	**0.61**	0.00	312.22	415
Ste36a	9526	9.07	0.01	12.02	0.00	6.51	0.00	11.92	**0.52**	**20.00**	13.04	18
Ste36b	15852	15.95	0.18	21.46	0.01	7.70	4.00	11.88	**0.01**	**96.00**	8.46	18
Ste36c	8239.11	7.24	0.00	9.46	0.00	6.12	0.00	13.25	**0.28**	**36.00**	14.08	18
Tai20a	703482	4.37	0.02	5.11	0.01	3.38	0.00	0.29	**0.60**	**12.00**	1.25	3
Tai25a	1167256	4.08	0.00	4.76	0.00	3.26	0.00	1.88	**0.96**	0.00	0.97	6
Tai30a	1818146	3.84	0.00	4.47	0.00	3.02	0.00	4.08	**1.55**	0.00	6.42	10
Tai35a	2422002	3.72	0.00	4.47	0.00	3.50	0.00	9.17	**1.73**	0.00	19.04	20
Tai40a	3139370	3.68	0.00	4.56	0.00	3.51	0.00	15.50	**2.54**	0.00	17.29	25
Tai50a	4941410	3.71	0.00	4.46	0.00	3.97	0.00	39.00	**3.00**	0.00	32.17	50
Tai60a	7208572	3.54	0.00	4.20	0.00	3.83	0.00	69.83	**3.06**	0.00	62.42	89
Tai80a	13557864	2.78	0.00	3.22	0.00	3.14	0.00	197.54	**2.54**	0.00	128.92	223
Tai100a	21125314	2.49	0.00	3.03	0.00	2.80	0.00	878.08	**2.37**	0.00	622.54	1000
Tho30	149936	3.72	0.03	4.60	0.01	3.34	0.00	5.92	**0.20**	**36.00**	5.62	10
Tho40	240516	3.70	0.00	4.46	0.00	2.97	0.00	18.63	**0.45**	0.00	19.58	25
Wil50	48816	1.37	0.00	1.65	0.00	1.07	0.00	91.50	**2.09**	0.00	73.96	120

The results show that this approach gets results consistently better than those obtained by the compared algorithms being, therefore, an attractive alternative to be implemented as part of metaheuristic algorithms, such as procedures of intensification of evolutionary algorithms.

Acknowledgment

This research was partially supported by CNPq and the ANP, Brazilian National Oil Agency, PRH-22 project.

References

1. Ahuja,R.K., Jha, K.C., Orlin, J.B., Sharma,D.: Very large-scale neighborhood search for the quadratic assignment problem. Informs Journal on Computing **19(4)** (2007) 646–657
2. Conover, W.J.: Practical Nonparametric Statistics, John Wiley & Sons (2001)
3. Drezner, Z.: The extended concentric tabu for the quadratic assignment problem. European Journal of Operational Research **160(2)** (2005) 416-422
4. Drezner, Z.: Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. Computers & Operations Research **35(3)** (2008) 717-736
5. Glover, F., Rego, C.: Ejection chain and filter-and-fan methods in combinatorial optimization. 4OR: A Quarterly Journal of Operations Research **4(4)** (2006) 263–296
6. de Kock, E.A., Aarts, E.H.L., Essink, G., Jansen, R.E.J., Korst, J.H.M.: A variable-depth search for the recursive bipartitioning of signal flow graphs. OR Spectrum **17(2-3)** (1995) 159–172
7. Koopmans, T.C., Beckmann,M.J.: Assignment problems and the location of economic activities. Econometrica **25** (1957) 53–76
8. Li, Y., Pardalos, P.M., Resende, M.G.C.: A greedy randomized adaptive search procedure for the quadratic assignment problem. In: Quadratic Assignment and Related Problems, Pardalos, P.M., Wolkowicz, H. (Eds.), DIMACS Series on Discrete Mathematics and Theoretical Computer Science **16** (1994) 237–261
9. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Operations Research **21** (1973) 498–516
10. Loiola, E.M., Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.M.: A survey for the quadratic assignment problem. European Journal of Operational Research **176(2)** (2007) 657–690
11. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. In: Handbook of Metaheuristics, Glover, F., Kochenberger, G.(Eds.),International Series in Operations Research & Management Science **57**, Kluwer Academic Publishers (2002) 321–353
12. Stützle, T.: Iterated local search for the quadratic assignment problem. European Journal of Operational Research **174** (2006) 1519-1539
13. Taillard, E.D.: Robust tabu search for the quadratic assingnment problem. Parallel Computing **17** (1991) 443–455
14. Yagiura,M., Yamaguchi, T., Ibaraki, T.: A variable depth search algorithm with branching search for the generalized assignment problem. Optimization Methods and Software **10(2)** (1998) 419–441