GRASP with Path-Relinking for the Generalized Quadratic Assignment Problem

Geraldo R. Mateus[*] Mauricio G.C. Resende [⋆] Ricardo M.A. Silva [◇]

[*] *Department of Computer Science, Federal University of Minas Gerais CEP 31270-010, Belo Horizonte, MG, Brazil*

[⋆] *Algorithms and Optimization Research Department, AT&T Labs Research 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.*

[◇] *Computational Intelligence and Optimization Group, Department of Computer Science, Federal University of Lavras C.P. 3037, CEP 37200-000, Lavras, MG, Brazil*

Abstract

The generalized quadratic assignment problem (GQAP) is a generalization of the NP-hard quadratic assignment problem (QAP) that allows multiple facilities to be assigned to a single location as long as the capacity of the location permits. In this paper, we propose a GRASP with path-relinking (GRASP-PR) for the GQAP. Experimental results illustrate the effectiveness of GRASP-PR on instances found in the literature.

Keywords: *GRASP, Path-Relinking, Generalized Quadratic Assignment Problem.*

1 Introduction

Let $N = \{1, \dots, n\}$ denote the set of facilities and $M = \{1, \dots, m\}$ the set of locations. Furthermore, denote by $A_{n\times n} = (a_{ii'})$ the flow between facilities $i \in N$ and $i' \in N$, such that $a_{ii'} \in \Re^+$ if $i \neq i'$ and otherwise $a_{ii'} = 0$, by $B_{m\times m} = (b_{jj'})$ the distance between locations $j \in M$ and $j' \in M$, such that $b_{jj'} \in \Re^+$ if $j \neq j'$ and otherwise $b_{jj'} = 0$, and by $C_{n\times m} = (c_{ij})$, the cost of assigning facility $i \in N$ to location $j \in M$, such that $c_{ij} \in \Re^+$. Let $z \in \Re^+$ be a scaling factor called the unit traffic cost, $q_i \in \Re^+$ be the capacity demanded by facility $i \in N$, and $Q_j \in \Re^+$, the capacity of location $j \in M$. The GQAP consists in finding $X_{n\times m} = (x_{ij})$, with $x_{ij} = \{0, 1\}$, where facility $i \in N$ is assigned to location $j \in M$ if and only if $x_{ij} = 1$, such that the constraints

$$\sum_{j\in M} x_{ij} = 1, \forall i \in N, \tag{1}$$

$$\sum_{i\in N} q_i x_{ij} \le Q_j, \forall j \in M, \tag{2}$$

$$x_{ij} \in \{0, 1\}, \forall i \in N, \forall j \in M$$

are satisfied and the objective function

$$\sum_{i\in N}\sum_{j\in M} c_{ij}x_{ij} + z\sum_{i\in N}\sum_{j\in M}\sum_{i'\in N, i'\neq i}\sum_{j'\in M} a_{ii'}b_{jj'}x_{ij}x_{i'j'}$$

is minimized. Constraints (1) guarantee that each facility is assigned to exactly one location, while constraints (2) ensure that location capacities are not violated.

In this paper, we present a GRASP with path-relinking [5, 7] heuristic for the generalized quadratic assignment problem. Extensive computational experiments on benchmark test problems show the effectiveness of these heuristics.

The paper is organized as follows. In Section 2, we describe the GRASP with path-relinking procedure. Computational results are described in Section 3.

2 GRASP with Path-relinking for GQAP

A GRASP is a multi-start heuristic where at each iteration a greedy randomized solution is constructed to be used as a starting solution for local search. Local search repeatedly substitutes the current solution by a better solution in the neighborhood of the current solution. Each such replacement is called a *move*. If there is no better solution in the neighborhood, the current solution is declared a local minimum and the search stops. The best local minimum found over all GRASP iterations is output as the solution. One way to incorporate memory into GRASP is with path-relinking (Glover [2]). In GRASP with path-relinking (Laguna and Martí [5]), an elite set of diverse good-quality solutions is maintained to be used during each GRASP iteration. After a solution is produced with greedy randomized construction and local search, that solution is combined with a randomly selected solution from the elite set using the path-relinking operator. The combined solution is a candidate for inclusion in the elite set and is added to the elite set if it meets certain quality and diversity criteria. Algorithm 1 shows pseudo-code for the GRASP with path-relinking heuristic for the GQAP.

The algorithm takes as input the set N of facilities, the set M of locations, the flow matrix A, the distance matrix B, the assignment cost matrix C, the scaling factor z, the facility demands $q_i,\ i \in N$, and the location capacities $Q_j,\ j \in M$, and outputs a feasible solution p^* specifying the location of each facility in the best solution found. After initializing the elite set P as empty, the GRASP with path-relinking iterations are computed until a stopping criterion is satisfied. This criterion could be, for example. a maximum number of iterations, a target solution quality, or a maximum number of iterations without improvement. During each iteration, a greedy randomized solution is generated and local search is applied using it as a starting point, resulting in a solution p. If the greedy randomized solution is infeasible, a feasible solution is randomly selected from the elite set and local search is applied on this solution. Path-relinking is applied between p and some elite solution q only if the elite set has at least a minimum number of elements. Otherwise, solution p is simply added to the elite set if it is sufficiently different from the solutions already in the elite set. To more precisely define the term *sufficiently different*, let the *symmetric difference* $\Delta(x, y)$ between two solutions x and y be defined as the number of moves needed to transform x into y or vice-verse. For a given level of difference δ, we say x is sufficiently different from all elite solutions in P if $\Delta(x, p) > \delta$ for all $p \in P$, which we indicate by the notation $x \not\approx P$. If the elite set is not yet full, the solution r resulting from path-relinking is added to the elite set if $r \not\approx P$. Otherwise, if r is not of worse quality than any elite solution and $r \not\approx P$, then it will be added to the elite set in place of some elite solution. Among all elite solutions having cost no better than that of r, the one most similar to r, i.e. with smallest symmetric difference with respect to r, is selected to be removed from the elite set. At the end, the best elite set solution is output as the solution of the GRASP with path-relinking heuristic.

2.1 Greedy randomized construction.

The construction procedure builds a solution one assignment at time. Suppose a partial solution is on hand, i.e. a number of assignments have already been made. To make the next assignment, the procedure needs to select a new facility and a location. Locations are made available, one at time. The procedure

```
Data  : $N, M, A, B, C, z, q_i, Q_j$.
Result: Feasible solution $p^*$.
$P \leftarrow \emptyset$;
while stopping criterion not satisfied do
    $p \leftarrow$ GreedyRandomized($\cdot$);
    if elite set $P$ has enough elements then
        if $p$ is not feasible then
            Randomly select a new solution $p \in P$;
        end
        $p \leftarrow$ LocalSearch($p$);
        Randomly select a solution $q \in P$
        $r \leftarrow$ PathRelinking($p, q$);
        if elite set $P$ is full then
            if $c(r) \leq \max\{c(s) \mid s \in P\}$ and $r \not\approx P$ then
                replace the element most similar to $r$ among all elements with cost
                worse than r;
            end

        else if $r \not\approx P$ then
            $P \leftarrow P \cup \{r\}$;
        end

    else if $p$ is feasible and $p \not\approx P$ then
        $P \leftarrow P \cup \{p\}$;
    end
end
return $p^* = $ argmin$\{c(s) \mid s \in P\}$;
```

Algorithm 1: A GRASP with path-relinking heuristic for the GQAP.

randomly determines whether to use a new location or a previously chosen location, favoring a new location when the previously chosen locations have insufficient or barely sufficient available capacity. If the procedure determines that a previously chosen location is to be selected, it then determines which facilities can be assigned to that location with the maximum available capacity and randomly selects one of these facilities to be assigned. Of the locations that can accommodate this facility, one is selected at random and the assignment is made. On the other hand, if there is no previously chosen location with sufficient capacity or if the available capacity is barely sufficient, a new location is selected at random from the set of yet unchosen locations. As the above procedure is not guaranteed to produce a feasible solution, it is repeated a maximum number of times until it either ends with a valid assignment or with an infeasible solution.

2.2 Approximate local search.

The construction procedure of Subsection 2.1 produces a feasible solution p that is not guaranteed to be locally optimal. A local search procedure is applied starting at p to find an approximate local minimum. The local search procedure makes use of two neighborhood structures which we call *1-move* and *2-move*. A solution in the 1-move neighborhood of p is obtained by changing one facility-to-location assignment in p. Likewise, a solution in the 2-move neighborhood of p is obtained by simultaneously changing two facility-to-location assignments in p.

One way to carry out a local search in these neighborhoods is to evaluate moves in the 1-move neighborhood and move to the first improving solution. If no 1-move improving solution exists, 2-move neighborhood solutions are evaluated and a move is made to the first improving solution. Another way to carry out the local search is to evaluate all 1-move and 2-move neighborhood solutions and move to the best improving solution. In both variants, the search is repeated until no improving solution in the neighborhoods exists. We propose a tradeoff approach here. Instead of evaluating all of the 1-move and 2-move neighborhood solutions, we sample these neighborhoods and populate a candidate list with improving solutions. One of the solutions from the candidate list is randomly selected and a move is made to that solution. The search is repeated until no improving solution is sampled. Because solutions are sampled, not all neighbors may be evaluated. Consequently, the best solution found may not be a local minimum. We call this solution an *approximate local minimum.*

2.3 Path-relinking

Motived by the fact that a single move from a solution x in the direction of a target solution x_t does not guarantee the feasibility of the new constructed solution, a new variant of path-relinking is proposed in this paper.

Suppose that among the differences between x and x_t is the location assigned to facility f. In other words, while the location assigned to f in x_t is l, the location assigned to f in x is l', with $l \neq l'$. In this case, is not necessarily feasible to perform a move in x that assigns f to l. If the capacity Q_l is not violated, then the new solution is feasible. Otherwise, a repair procedure must be applied to try to make it feasible.

In this repair procedure, a facility set F is created with all not yet fixed facilities assigned to location l for which capacity is violated. Next, the set $T \subseteq F$ is constructed with all facilities in F having demands less than or equal to the maximum available capacity of locations in M. After a facility from T is randomly selected, set R consists of locations in M that can accommodate it. A location is selected from set R and the facility is assigned to it. This process is repeated until the capacity of location l has a nonnegative slack.

The path-relinking process is a sequence of steps from x_s to x_t. In each step a move is performed from the current solution x with or without repair. Next, a facility i is randomly selected from a set composed of all not yet fixed facilities corrected in the step. A facility is *corrected* when its location becomes the same as the one assigned to it in the target solution x_t. After facility i is fixed, the next step begins. This process continues until the target solution x_t is reached or when no feasible solution is obtained from x.

This path-relinking is different from the standard variant because given solutions x_s and x_t, their commons elements are not kept fixed a priori, such that a small portion of the solution space spanned by the remaining elements is explored. The new variant fixes one facility at time at each step.

3 Experimental results

In this section, we report the experimental results using the GRASP-PR heuristic introduced in this paper. First, we describe our test environment. Next, we compare our implementation of GRASP-PR with other heuristics from the literature on a suite of test problems.

3.1 Test environment.

All experiments with GRASP-PR were run on a Dell PE1950 computer with dual quad core processors and 16 Gb of memory, running Red Hat Linux version 5.1.19.6 (CentOS release 5.2, kernel $2.6.18{-}53.1.21.el5$). GRASP-PR was implemented in Java and compiled into bytecode with javac version 1.6.0_05. The random-number generator is a implementation of the Mersenne Twister algorithm (Matsumoto and Nishimura [8]) from the COLT library. COLT is a open source library for high performance scientific and technical computing in Java. See `http://acs.lbl.gov/~hoschek/colt/`.

3.2 Comparing GRASP-PR with other algorithms

In the experiments to follow, we made use of the test problems used by Lee and Ma [6], Cordeau et al. [1], and Hahn et al. [4]. Among the largest instances are those created by Cordeau et al. [1] with 20 to 50 facilities and 6 to 20 locations. Cordeau et al. developed a memetic algorithm which found promising results for these larger instances. Table 1 compares these results with ones obtained with the GRASP-PR heuristic.

Table 1: Summary of results for memetic and GRASP-PR heuristics on the Cordeau et al. [1] instances. Times are given in seconds.

	Memetic algorithm		GRASP-PR times				
instance	solution	time	min	max	avg	s.d.	0.95
20-15-35	**1471896**	96.0	0.16	38.87	7.09	6.48	21.04
20-15-55	**1723638**	102.0	0.24	14.42	2.88	2.18	7.69
20-15-75	**1953188**	102.0	0.26	12.82	2.02	1.72	5.25
30-06-95	**5160920**	114.0	0.55	23.81	2.60	2.22	6.44
30-07-75	**4383923**	156.0	0.50	38.47	7.86	5.47	18.18
30-08-55	**3501695**	96.0	0.18	4.89	1.62	0.95	3.60
30-10-65	**3620959**	210.0	2.75	1032.80	122.55	146.54	514.82
30-20-35	**3379359**	564.0	1.08	4441.40	79.43	314.14	166.21
30-20-55	**3593105**	462.0	1.28	150.11	25.28	21.22	66.82
30-20-75	**4050938**	522.0	2.11	759.81	41.64	68.66	148.43
30-20-95	5726530	2652.0	237.02	638983.5	55580.31	118661.79	160232.36
35-15-35	**4456670**	456.0	8.41	1717.94	307.65	242.74	775.25
35-15-55	**4639128**	384.0	4.33	75.69	21.24	11.92	42.47
35-15-75	**6301723**	396.0	5.18	621.83	68.57	74.39	183.19
35-15-95	**6670264**	864.0	14.40	18470.45	1551.78	3533.59	7356.88
40-07-75	**7405793**	180.0	4.53	377.06	59.66	51.29	159.00
40-09-95	**7667719**	1140.0	6.18	5017.56	419.08	612.62	1490.31
40-10-65	**7265559**	240.0	0.84	115.06	17.96	15.90	52.73
50-10-65	**10513029**	504.0	2.52	84.64	24.68	16.33	64.04
50-10-75	**11217503**	606.0	22.79	24507.34	1384.36	3499.53	4404.50
50-10-95	**12845598**	1254.0	9.97	1059.59	89.36	91.95	200.20

The first column of Table 1 shows the instance names. These instances are labeled with three numbers: $n = |N|$, $m = |M|$, and a parameter $s \in \{1, \ldots, 100\}$ which controls the tightness of the capacity constraints.

The third column lists the time (in seconds) reported by Cordeau et al. [1] for their memetic algorithm to find the solutions whose values are given in the second column. Boldface indicates the best solutions found by the memetic algorithm. For instance 30-20-95, Cordeau et al. [1] also report a better value of 5710645.

For each instance, we made 200 independent runs of GRASP-PR, with the exception of instances 30-20-95, 35-15-95, and 50-10-75 for which we made only 30, 59, and 120 runs, respectively. Each run stopped when a solution value as good as the one in column 2 was found. The last columns denote the minimum, maximum, and average times, and standard deviation of these runs to find solutions with values equal to those of column 2. Finally, column 8 lists the time for 95% of these runs to find solutions having those same target values.

GRASP-PR has achieved the target values on all instances in Table 1, with an average performance improvement varying between a factor of 1.482 and 59.186 with respect to the memetic algorithm, except for instances 30-20-95, 35-15-95, and 50-10-75. The GRASP-PR heuristic found the best solution found by the memetic algorithm (5710645) for instance 30-20-95 in 161,214.86 seconds. Furthermore, a better performance was verified on all instances of Lee and Ma [6] and Hahn et al. [4], with an average improvement factor of 11.2 to 1004.6 and 16.8 to 291.1, respectively. Due to space limitations, we do not show those experiments in detail.

4 Concluding remarks

In this paper, we propose a GRASP with a novel approximate local search and path-relinking approach for the generalized quadratic assignment problem. The algorithm was implemented in Java and extensively tested. Computational results from several instances from the literature demonstrate that the heuristic is a well-suited approach for GQAP.

Acknowledgement

Ricardo M. A. Silva was supported in part by the Brazilian National Science and Technology Council (CNPq). This work was done while he was a post-doc at AT&T Labs Research.

References

[1] J.-F. Cordeau, M. Gaudioso, G. Laporte, and L. Moccia, *A memetic heuristic for the generalized quadratic assignment problem*, INFORMS J. on Computing **18** (2006) pp. 433–443.

[2] F. Glover., *Tabu search and adaptive memory programing – Advances, applications and challenges*, Interfaces in Computer Science and Operations Research, R.S. Barr, R.V. Helgason, and J.L. Kennington, eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Kluwer, (1996), pp. 1–75.

[3] F. Glover, M. Laguna, and R. Martí., *Fundamentals of scatter search and path relinking*, Control and Cybernetics **39** (2000) pp. 653–684.

[4] P.M. Hahn and B.-J. Kim, M. Guignard, J. MacGregor Smith, and Y.-R Zhu, *An algorithm for the generalized quadratic assignment problem*, Computational Optimization and Applications **40** (2008) pp. 351–372.

[5] M. Laguna and R. Martí, *GRASP and path relinking for 2-layer straight line crossing minimization*, INFORMS Journal on Computing, **11** (1999) pp. 44–52.

[6] C.-G. Lee and Z. Ma, *The generalized quadratic assignment problem*, MIEOR TR2005-01, Department of Mechanical and Industrial Engineering at the University of Toronto (2005).

[7] M.G.C. Resende and C.C. Ribeiro, *GRASP with path-relinking: Recent advances and applications*, in *Metaheuristics: Progress as Real Problem Solvers*, T. Ibaraki and K. Nonobe and M. Yagiura, eds., Springer, 2005, pp. 29–63

[8] M. Matsumoto and T. Nishimura, *Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*, ACM Transactions on Modeling and Computer Simulation **8** (1998) pp. 3–30.