© International Journal of Combinatorial Optimization Problems and Informatics, Vol. 8, No. 3, Jan-April 2017, pp. 33-38. ISSN: 2007-1558.

Quadratic Assignation Problem: A solution approach with parallel GRASP

Beatriz Bernábe Loranca, Martín Estrada Analco, Rogelio González Velázquez, Abraham Sánchez López, Jorge Cerezo Sánchez, Mario Bustillo Díaz
Benemérita Universidad Autónoma de Puebla, Universidad Tecnológica de Puebla
beatriz.bernabe@gmail.com

Abstract. The goal of this work is to establish and solve the Quadratic As-signation Problem (QAP) as a combinatory optimization problem by means of GRASP (Greedy Randomized Adaptive Search Procedure) as an approximation method to QAP.Applying GRASP to QAP produces good results to obtain solutions close to the optimum or even reach the optimum in several cases. The implementation of a sequential program was successfully made in C. The robustness of GRASP obeys to the inclusions of strategic procedures for each one of the three local search neighborhood structures employed as a second phase of GRASP. Finally a parallel system was built to reduce the time cost of the CPU.

Keywords: GRASP, QAP, Parallel system.

1 Introduction

The Quadratic Assignation Problem was originally proposed in 1957 by Koopmans and Beckman, analyzing the location of economic activities [1, 2]. The QAP is a classic problem in combinatory optimization and basically consists in finding the optimal assignation of *n* facilities to *n* locations with the end of minimizing transportation costs. A distance matrix, representing the distances between the locations and the respective flow of materials between them, is the input data.

The solution space of QAP is of size *n*! On the other hand, it has been proven that it belongs to the NP-complete class [3] which implies that an exact algorithm that can solve it in a reasonable amount of time, doesn't exist, even for instances of moderate size. At this point it is necessary to design approximation algorithms such as metaheuristics before the inefficiency of exact methods. With the arrival of metaheuristics such as Tabu Search, Simulated Annealing and GRASP among others, great improvements have been made to find high quality solutions in complex problems. In this work we have applied GRASP to solve QAP.

To test our algorithm we have used instances from QAPLIB, where it is possible to retrieve dissertations, articles, results and test problems related with QAP (http://www.imm.dtu.dk/~sk/qaplib/#address) [4].

2 Problem Description

Let $N = \{1, 2, 3, ..., n\}$ and $F = (f_{ij})$ and $D = (d_{ij})$ be two square symmetric matrices of *n* x *n* size; we try to find the assignation of *n* facilities to *n* locations, this is to say, a permutation $P \in \Pi_N$ that minimizes the objective function

$$z = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{P(i)P(j)}$$

Where Π_N is the set of all the permutations of the set N, $f_{i,j}$, represents the flow of materials from facility to facility and $d_{i,j}$ is the distance from location to location .Working details must be given concisely; well-known operations should not be described in detail.

Received May 24, 2017 / Accepted Jun 11, 2017

Multiple applications of QAP have been found in areas as diverse as engineering and logistics. Some of the problems treated are the total minimization of the wiring of electronic circuits, the optimal placement of industrial facilities, task scheduling, distribution of medical services within a hospital and the placement of trailers over rail plat-forms in an intermodal transportation environment.

Within the requirements for the solution of QAP, a data matrix is used. In practice this is known as compact matrix. Assume that the flow and distances matrices and are symmetric, then we can write the instances of the data into one single matrix that compacts and as follows:

$$C=\begin{bmatrix} 0 & d_{12} & d_{13} & d_{14} & \cdots & d_{1n} \\ f_{21} & 0 & d_{23} & d_{24} & \cdots & d_{2n} \\ f_{31} & f_{32} & 0 & d_{34} & \cdots & d_{3n} \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ & & & & 0 & d_{n-1\,n} \\ f_{n1} & f_{n2} & \cdots & \cdots & f_{n\,n-1} & 0 \end{bmatrix}$$

3 Development

GRASP is a metaheuristic that consists of two phases, the first phase builds a solution whereas the second one is for post-processing. These could be presented in tabular or graph form, with appropriate statistical evaluation.

First Phase (constructive GRASP)

Stage 1, a candidate list is generated, which has been previously bounded by two parameters, one of the candidates is randomly chosen from which the first two assignations are made,

Stage 2, one by one the rest of the *n-2* assignations are added according to a greedy function, wherewith having completed the solution produces a feasible solution that is expected to be in the neighborhood of an optimal solution or at least a solution very close to the optimum, with this ends the first phase or constructive phase.

Second Phase (Post-processing GRASP)

The second phase or improvement phase is started taking the solution originated from the first phase as the initial solution, this is done through the use of some local search procedure, which once is completed we'll have an optimum local solution, which could be as well a global optimum.

For the development of QAP, we have two phases (construction and post-processing) that are described below:

3.1 Constructive phase for QAP

Stage 1. The two initial assignations are done simultaneously, specifically it is indicated that the resource i is assigned to location k and resource j is assigned to location l, while the cost corresponding to this assignation pair is $f_{ij} \cdot d_{kl}$.

Let α and β, $(0 < \alpha, \beta < 1 < 0)$ be the parameters that restrict the candidate list. Let $F = (f_{ij})$ and $D = (d_{kl})$ be the input symmetric matrices *n x n* with 0 in the diagonal wherewith the non-symmetric square compact matrix is formed.

$$C=\begin{bmatrix} 0 & d_{12} & d_{13} & d_{14} & \cdots & d_{1n} \\ f_{21} & 0 & d_{23} & d_{24} & \cdots & d_{2n} \\ f_{31} & f & 0 & d_{34} & \cdots & d_{3n} \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ & & & & 0 & d_{n-1\,n} \\ f_{n1} & f_{n2} & \cdots & \cdots & f_{n\,n-1} & 0 \end{bmatrix}$$

It is specified that [x] is the integer part of x. Let $m = n\,(n\text{-}1)/2$ be the number of inputs in the upper and lower triangles of the compact matrix. Afterwards these distances and flows inputs are listed in increasing and decreasing order respectively, this is,

$$d_{k_1 l_1} \leq d_{k_2 l_2} \leq \cdots \leq d_{k_m l_m}$$

$$f_{i_1 j_1} \geq f_{i_2 j_2} \geq \cdots \geq f_{i_m j_m}$$

Now we have two ordered lists, and the parameter β is used to restrict both lists, thus they are cut up to the element [β*m*]. A new list is generated which stores the product of the distances by the flows in the corresponding order, this is how the new list is obtained

$$f_{i_1 j_1} \cdot d_{k_1 l_1},\ f_{i_2 j_2} \cdot d_{k_2 l_2},\ \cdots,\ f_{i_{[\beta m]} j_{[\beta m]}} \cdot d_{k_{[\beta m]} l_{[\beta m]}}$$

The last list is ordered in increasing order and we use parameter α to get the final restricted candidate list (RCL) from which only the first $[\alpha \cdot \beta \cdot m]$ elements will be taken and element $f_{ij} \cdot d_{kl}$ will be randomly chosen that represents the cost of committing the assignation pair (i,k) and (j,l), this is to say that we have two components of the solution, which to simplify will be written as permutation, where the *k-th* component and the *l-th* component are placed. Here we can appreciate the random component of the method. With this the first constructive stage concludes.

Stage 2. In this stage we seek to complete the initial solution by calculating the *n*-2 remaining assignations, by mean of a greedy procedure that generates one by one the assignations with the minimum cost according to the existent assignations and ties are randomly broken. This stage is supported by an adaptive component which is in charge of updating the solution as it is being built up**.**

Let $\Gamma = \{(j_1, l_1), (j_2, l_2), ..., (i_r, l_r)\}$ be the set of assignations being built. Stage 2 begins with $|\Gamma| = 2$ as a result of stage 1. Let $C_{ik} = \sum_{(j,l)\in\Gamma} f_{ij} \cdot d_{kl}$ be the cost of assigning facility *i* to location *k* according to the existent assignations. We select the couples (*i, k*) not assigned that have the minimum cost C_{ik}, this is the core of the greedy procedure. In this part there is as well a restricted candidate list, where the costs C_{ik} are ordered in increasing order and one of the first pairs $[\alpha \cdot z]$ is randomly chosen where z is the amount of pairs not assigned another random component.

The adaptive component of GRASP has the function of updating the set Γ by adding new assigned pairs, this is $\Gamma = \Gamma \cup \{(i, k)\}$

At the end of this stage, the first stage ends as well. A solutions has been built contained in the set $\Gamma = \{(j_1,l_1),(j_2,l_2),\ldots,(j_n,l_n)\}$ ordering the first components of the pairs, we take the second components to form the permutation equal to the solution. In summary we have a good quality solution to begin with.

```
Procedure stage2 (α,(j1,l1),(j2,l2))
Γ= {(j1,l1),(j2,l2)};
While |Γ|≤ n   do
z = 0;
    for i = 1 to n;
         for j = 1 to n;
             if  (i,j)∉ Γ then
                              Cik = Σ(i,j)∈Γ fij dkl ;
                              inheap (Cik );
                              z = z +1;
            end{if};
       end{for};
   end{for};
   s = random(1,..., [αz]);
   Cik = outheap(s)
   Γ = Γ ∪ {(i,k)};
end{while};
end{estapa2};
```

Algorithm for the stage 2 of the constructive phase of GRASP

3.2 Post-processing phase for QAP

This phase pursues the mission of improving the solution produced during the constructive phase. In our case we will apply a local search procedure with three neighborhood structures; 2-exchange, λ-exchange and Nstar. The structures λ–exchange and Nstar are more complex structures based on detecting good moves to avoid executing them. For example if (*i*, *j*) is a good move in the permutation, λ-exchange keeps the elements corresponding to *i* and *j* fixed allowing only the exchange of the remaining elements of the permutation, whereas Nstar allows moving the elements corresponding to *i* and *j* with the rest of the elements, but forbids the exchange between them.

4 Conclusions

The results of the program were obtained by running an executable program, generated by the compiler gcc (v2.7) in a machine SGI/CRAY ORGIN 2000 located at the DGESCA UNAM with 32 processors R 10000 at 195 Mhz, with IRIS 6.4 OS [5].
The test instances were obtained from the QAPLIB website [4], twelve proposals by Nugent which dimensions are: 5, 6, 7, 8, 12, 15, 20, 21, 22, 24, 25 and 30 and three by Skorin-Kapov which dimensions are 42, 64 and 81.
All the results shown were obtained by restricting the candidate list with the parameters $\alpha = 0.5$ and $\beta = 0.1$ which were determined experimentally. For the instances which dimension is less or equal to 42, the stopping criterion used was the following: If the best value found hasn't been improved after n^2 iterations, then finalize and return the best value; and for the last two instances the stopping criterion was a fixed number of $4n$ iterations, these stopping criteria were obtained experimentally.
The statistic values in the tables were obtained from 20 runs of the sequential program, for each instance in every neighborhood structure design 2-exchange, λ-exchange and Nstar (N*).

Table 1. Nomenclature

Problem	*n*	OVBVK	BVF	LB	EPLB	OSP
Nug5	5	25	25	25	0	100
Nug6	6	43	43	43	0	100
Nug7	7	74	74	74	0	100
Nug8	8	107	107	97	9.34%	100
Nug12	12	289	289	264	8.65%	90
Nug15	15	575	575	542	5.73%	100
Nug20	20	1285	1285	1119	12.91%	100
Nug21	21	1219	1219	1004	17.63%	35
Nug22	22	1798	1798	1417	21.19%	100
Nug24	24	1744	1744	1419	18.63%	95
Nug25	25	1872	1872	1532	18.16%	70
Nug30	30	3062	3062	2886	5.75%	10
Sko42	42	7906	7926	7467	5.79%	0
Sko64	64	24249	24436	22868	6.41%	0
Sko81	81	45499	45738	43036	5.90%	0

Table 1 shows the nomenclature for the parameter:
n: The dimension of the compact matrix,
OVBVK: Optimal Value or Best Value Known,
BVF: Best Value Found,

LB: Lower Bound [6],
EPLB: Error Percentage where with BVF exceeds the lower bound,
OSP: Percentage in which the optimal solution or the best value known was obtained,
IAGRASP: Iterations Average of GRASP in which the OVBVK was obtained for the 3 strategies (λ, neighborhood structure 2 and N*),
ACPUT: Average CPU Time.
Table 2 shows the results with previous nomenclature for the parameters from table 1:
The numbers in the cells of column OVBVK are reported in the literature as the optimal values from $n = 5$ until $n = 25$ and for dimensions 30, 42, 64 and 81 as the best values known (several sources agree on this point). The numbers in column LB are considered from the QAPLP Statistics table, which was generated with an interior points algorithm from linear programming. Whereas the values in column EPLB are calculated from the formula EPLB = [(BVF - LB)/BVF]* 100%. The following figure reveals the behavior of the performance of the 3 neighborhood structures.
In the figure 1, is possible to observe the behavior of the results obtained by the sequential program for the neighborhood structures 2-exchange, l-exchange and N*, with respect to IAGRASP.
We can see that l-exchange is the best proposal regarding the number of iterations and 2-exchange is the best option with regard to the CPU time and as well as the number of times the optimum is reached.

Table 2. Results

IAGRASP (Nstar)	ACPUT (Nstar)	IAGRASP (λ)	ACPUT (λ)	IAGRASP (neighborhood structure 2)	ACPUT (neighbohood structure 2)
3	0	1	0	2	0
4	0	4	0	3	0
3	0	5	0	5	0
4	0	5	0	4	0
55	0.87	55	0.81	65	0.23
121	6.1	56	2.65	67	0.79
15	6.59	77	14.21	73	3.61
428	117.2	359	99.87	485	32.97
215	82	101	34.82	250	24.05
408	230.14	241	117.35	435	63.89
270	217.35	339	205.3	420	56.65
1647	2720.32	363	561.18	529	224.78
3263	28186.3	2203	20171.2	3823	9294.33
256	5262.38	256	4858.48	256	5989.71
324	23451.2	324	21752.9	324	110638.2

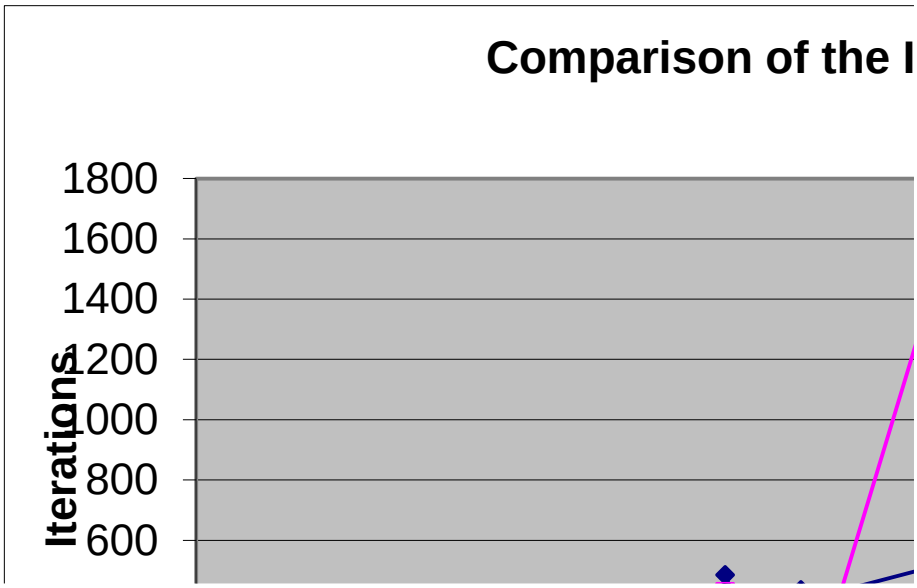

Fig. 1. Performance tests for the 3 neighborhood structures 2-exchange, λ-exchange and Nstar

References

1. Li, Y., Pardalos, P.M., Resende, M.G.: A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. In P.M. Pardalos and H. Wolkowicz, editors, Quadratic assignment and related problems, vol. 16 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pp 237-261. American Mathematical Society, 1994.
2. Feo, T., Resende, M.G.:. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, vol 6, pp109-133, 1995.
3. Sahni S., Gonzalez T.: P-complete approximations problems. J. Asssoc. Comp. Machine. vol. 23, pp 555-565, 1976.
4. Burkard, R. E., Karisch, S.E., Rendl F.:QAPLIB - A Quadratic Assignment Problem, Library, http://www.imm.dtu.dk/~sk/qaplib/ins.html.
5. Pardalos, P.M., Crouse, A.: A parallel algorithm for the quadratic assignment problem. In Proceedings of the supercomputing 1989 Conference, pp 351-360, ACM Press, 1989.
6. Resende, M.G., K.G., Drenzner, R.Z.: Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. Operations Research, vol 43,N° 5, september-october1995.