Solving Quadratic Assignment Problem (QAP) On Cluster with a Bound of Reformulation Linearization Techniques*

A.Djerrah[1] S.Jafar[2] V.D Cung[3]
and P.Hahn[4]

[1]Laboratoire PRiSM-CNRS, Université de Versailles-Saint Quentin en Yvelines, France
Emails: `Abdelaziz.Djerrah@prism.uvsq.fr`

[2] Laboratoire Informatique et Distribution, Grenoble, France,
Emails: `Samir.jafar@imag.fr`

[3] GILCO lab., ENSGI-INPG, Grenoble, France,
Emails: `Van-Dat.Cung@gilco.inpg.fr`

[4] University of Pennsylvania
Emails: `hahn@seas.upenn.edu`

Abstract

In this paper, we propose an parallel branch-and-bound algorithm for the quadratic assignment problem based upon an efficient lower bound calculation developed by Hahn and Grant [7]. The sequential version of this algorithm is very easy to achieve with the Bob++ library [1] and the results are very encouraging. For the parallel implementation, we ported our Bob++ library to the high level programming environment Athapascan [2]. The performances obtaining with this algorithm running on one cluster for the some instances of qap (size < 25) taken from the QAPLIB[3] are competitive.

Keywords. *Optimization, Quadratic Assignment Problem, Exact Algorithms, Branch-and-Bound, Lower Bound, Linearization, Cluster.*

1 Introduction

These last years, large and unsolved combinatorial optimization problems have been solved exactly. The first reason of this success is the progression in the research of

*This work is supported by two projects, ACI-GRID DOC-G and e-Toile of the French Research Ministry

[1]http://www.prism.uvsq.fr/~blec/Research/BOBO/

[2]http://www-id.imag.fr/Logiciels/ath1/

[3]http://www.opt.math.tu-graz.ac.at/qaplib/

lower bound of quality for these problems and the second reason is the progression of the calculation power by using *grid computing*.

The QAP belongs to the class of NP-hard problems and is considered as one of the most difficult. Because of the extreme difficulty of the problem, exact solution methods have often been implemented on high-performance computers. In the past ten years, a number of landmark computational results for the problem have been obtained using parallel processing hardware.

Two recent developments have resulted in a large improvement in the ability to solve QAPs exactly. The first is the dual procedure (DP) bound of Hahn and Grant [7], which derives from a level_1 reformulation linearization techniques (RLT) of the QAP. The second is the quadratic programming (QP) bound of Anstreicher and Brixius [3]. It is the speed of calculation of this QP bound and its parallel implementation that makes it effective in solving the most difficult problems to date. These two methods have made it possible to solve exactly heretofore-unsolved problems of size 30 [4, 8].

The scope of the work is the design of an parallel branch-and-bound algorithm based on the Hahn and Grant lower bound [7] procedure (DP) which has roots in the Hungrian algorithm this that never was done before. We are the first to propose it. This work allows us not only to resolve large instances of QAP but also to verify the results of Anstreicher and al. Since the publication of their results on the resolution of the Nugent 30, no verification is done.

The paper is organized as follows. In the second section, we give a brief description of the problem (definition, formulations and the different resolution approaches). We focus on the most critical formulation level_1 RLT of Adams and Johnson[2]. In the third section, we present the used branch-and-bound method and some lower bounds. In section four, we review the Hahn and Grant procedure (DP) derived from the formulation level_1 RLT. Section five consists to present our algorithm and different components used for solving the QAP on clutser. The current version of the code can execute itself on grid computing. The performance of the proposed algorithm on cluster is presented in section six. A set of problem instances is considered and benchmark results are given. Finally, concluding remarks are given in section 7.

1.1 Quadratic Assignment Problem:

The QAP was formulated by Koopmans and Beckmann (1957)[9] as the task to find for three $n \times n$ matrices $A = (a_{ik}), B = (b_{jl})$ and $C = (c_{ij})$ a permutation π of $\{1, ..., n\}$ that minimizes

$$Min \sum_{i,k=1}^{n} a_{ik} b_{\pi(i)\pi(k)} + \sum_{i=1}^{n} c_{i\pi(i)}.$$

The QAP can be used to formulate a variety of interesting problems in location theory, manufacturing, data analysis, and other areas. Unfortunately, the problem is typically for its size extraordinarily difficult to solve.

Linearization: When dealing with QAPs, it seems that the quadratic form in its objective function hopelessly destroys every attempt to efficiently solve the problem. One of the first ideas to avoid this inconvenient feature of the problem was probably the so called *linearization of the QAP.* That is, getting rid of the quadratic form in the QAP objective function by finding an equivalent linear formulation. A more or less complete list of references to various QAP linearizations can be found in [10]. Moreover, some of them are described in detail in [6]. Here, we present only the level_1 formulation of Adams and Johnson [2].

In 1990, Sherali and Adams[11] proposed a general strategy for linearizing 0-1 quadratic programming problems. When applied to the QAP, Adams and Johnson[2] proposed a new formulation for the QAP called level_1 formulation RLT. They define a new variable $y_{ijkl} = x_{ij}x_{kl}$.

$$(QAP :) Min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1k\neq i}^{n}\sum_{l=1l\neq j}^{n} C_{ijkl}y_{ijkl} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\sum_{k=1k\neq i}^{n} y_{ijkl} = y_{ijij} = x_{ij} = 0,1 \qquad \forall(i,j,l), l \neq j \tag{1}$$

$$\sum_{l=1l\neq j}^{n} y_{ijkl} = y_{ijij} = x_{ij} = 0,1 \qquad \forall(i,j,k), k \neq i \tag{2}$$

$$y_{ijkl} = y_{klij} \qquad \forall(i,j,k,l), k \succ i, l \neq j \tag{3}$$

$$\sum_{j=1}^{n} y_{ijij} = 1 \qquad \forall i = 1, ..., n \tag{4}$$

$$\sum_{i=1}^{n} y_{ijij} = 1 \qquad \forall j = 1, ..., n \tag{5}$$

$$y_{ijkl} \geq 0 \qquad \forall(i,j,k,l), k \neq i, l \neq j \tag{6}$$

In general, QAP linearizations based on MILP models present a huge number of variables and constraints, however they lead to the achievement of good lower bounds. The optimal solution for an MILP formulation is a lower bound for the corresponding QAP and dual solutions for the linear program of the MILP formulation is also a lower bound for the QAP (though not quite as good). We refer you to the references cited in Hahn and Grant [7] for discussion of the many lower bounds that fall in this category.

Lower bound: Notable among the bounds based on MILP relaxations is the dual ascent procedure bound of Hahn and Grant. This bound is based upon a level_1 reformulation linearization technique (RLT) suggested by Adams and Sherali [1]. It is characterized by the exploitation of a fact that other researchers of MILP relaxations relatively ignored, i.e., that certain variables were inexorably tied together. These variables came in "complementary" pairs that, due to the nature of the constraints, had to be equal. Hahn and Grant developed a computational method that exploited this fact, resulting in an iterative sequence that, at each iteration, produced a revised problem that was completely equivalent to the original problem. At the same time, this procedure produced a sequence of non-decreasing bounds. The resulting bounds are competitive in quality, when compared to some to the best bounds, and moreover are better than those bounds in computational time.

1.2 Branch-and-bound implementation

Among the exact solution methods for the QAP only branch-and-bound algorithms have proved to be useful.

It is impossible to get an accurate estimate of the work involved in executing brach-and-bound algorithm without carrying out the actual computations. Therefore, the only way to achieve an equitable division of the work among the various processes is to divide the worke as it is generated, i.e., dynamically during execution.

In order to divide the work among the processes some unit of works as well as some mechanism for dividing these units among the processes have to be defined. The unit of work can be arbitrary chosen. Examples of units of work are the branching from a single subproblem [Trienekens 1989, Kindervater and Trienekens 1988], the optimal solving of a subproblem [Kindervater 1989], or the computing of lower bounds to a subproblem generated by decomposition [Kindervater 1989].

The basic unit of work in our implementation correspond to branching from a single subproblem including the computation of the lower bounds on the subproblems thus generated. In our code, this unit work correspond to *QAPGenChild::operator()-(BobQAPNode *p)* method. This method take as argument the subproblem *BobQAPNode *p* (see Figure 1).

The master takes all important decisions: maintains the single central pool containing the active set and decides which subproblems to branch from and when.

Each time a slave process becomes, it extracts the subproblem ($P0$) from the pool (Figure2) according to the selection rule. Using the polytomic branching strategie, the current subproblem can be separated on many subproblems ($p0_0, p0_1, ..., p0_k$). Each new subproblem ($p0_i$)is evaluated by the *BobQAP::EvalNode()* function. Three situations presents:

First, the lower bound is smaller than the current best know solution, then the subproblem ($p0_i$) is added in the pool by *BobQAPAlgo::Search(p_i)*. Second, the subproblem is a solution, then the master process update the best know solution by the *SetSol()* method and third the subproblem is not feasible then it is deleted.

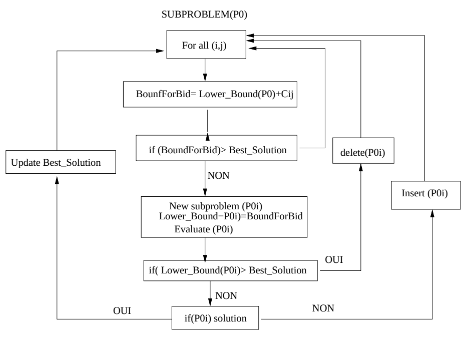

Figure 1: Unit of work

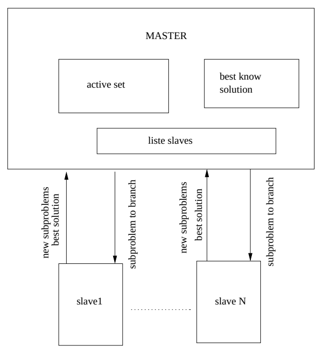

Figure 2: Parallel B&B

In the case where the best solution is updated, the master processe prunes the pool by applying the elimination rule.

2 Computational results

Our parallel algorithm has been developed in the programming language C++ and Athapascan . Out experiments will be run on i-cluster2 (104 nodes interconnected by a Myrinet network. Each node features 2 Itanium-2 processors 64 bits at 900 Mhz, 3 Gigabytes of memory and 72 gigabytes of local disk). This cluster use OAR [4] for reserve nodes in two fashions: interactif and passive. The cluster OS is RedHat Linux Advanced Server release 3.0.

In our preliminary experiments we run our tests using 14 processors. We execute each instance three times. The best know results and the result obtained by our algorithm are presented in Tables 1 and 2. The execution times, in second, required to compute the lower bound for each instance with differents algorithms are presented in Table 1.

For comparaison, we include, the level_1 RLT interior point calculation of Resende et.al.[?] and the level_1 RLT bounds of Hahn and Grant[7] . For each problem, we

[4]http://ita.imag.fr

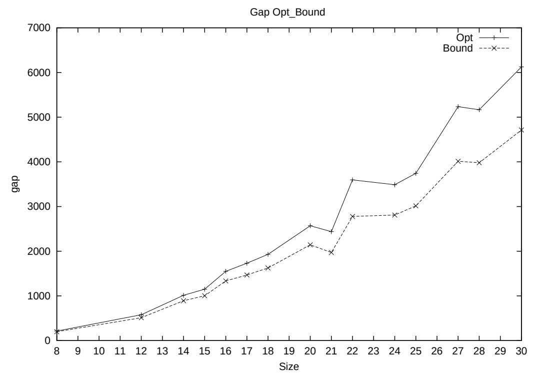

Figure 3: Geep Optimum-root

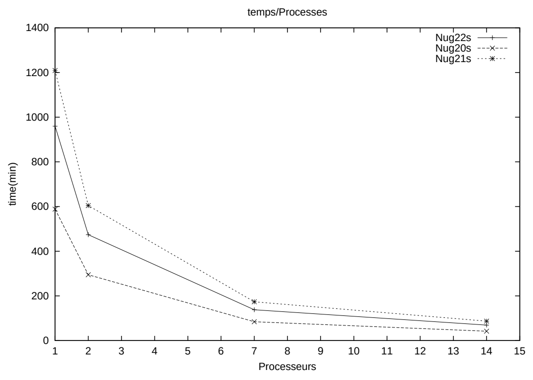

Figure 4: Performances of B&B

give the lower bound value in the root and the time required to the calculation. For the Hahn and Grant results, the bounds are obtained after 2000 iteration on a SPARC 10 workstation with a 75 MHz SuperSparc processor.

The figure 4 and Table 2 show the performances of our parallel branch-and-bound algorithm. Notice that we have very good performances. For some instances as the Nugent 18 and Nugent 20, we obtain super efficient greater than 100%.

Instance	opt	Resende et.al.	Hahn-and-Grant		Our Algo		
			bound	time (s)	bound	time (s)	iteration
nug12	578	523	523	82.5	512	0.393	148
nug15	1150	1041	1039	325.3	1003	1.563	238
nug20	2570	2182	2179	1457.1	2142	6.585	300

Table 1: *Lower bounds calculations.*

Instance	Nug18	Nug20	Nug21	Nug22
NP=1	27.12	294.23	604.45	479.85
NP=7	3.77	42.145	86.765	69.07
NP=14	1.69	20.845	43.57	34.725
$S(14)$	16.04	14.11	13.87	13.81
$e(14)$	114.57	100.78	99.07	98.64

Table 2: *Performances of parallel B&B algorithm*

3 Conclusion

In this paper, we presented an parallel branch-and-bound algorithm for solving the quadratic assignment problem on cluster. This algorithm is based on the Hahn and

Grant lower bound procedure (DP). Our algorithm was compared with the most actual algorithms: Hahn-and-Grant and the Ansetreicher-and-al. algorithms.

The algorithm proposed presented good speedups that encourages us to pass from cluster to grid.

The passage from cluster to grid is done without rewriting the code with Athapascan, and therefore very easy. Let us note that the experiences showed coming from the project e-Toile with more than 80 machines showed an effectiveness between 70% and 80%. The tests on grid are in realization course for some large instances.

Actually, we are working on the improving the lower bound and optimizing the data structures. We study the implementation of the lower bound based on the level_2 RLT of Hightower and Hahn. The preliminary sequential results are very encouraging. We think that this bound can reduce substantially the number of nodes on the branch-and-bound tree that need to be scanned. Though the solution time for computing those bounds is significantly greater than the time needed to compute the classical lower bounds.

An other way to explore is the parallelism on high level of the branch-and-bound algorithm. For each subproblm, we try to compute the lower bound in parallele while separating dual procedure on many independants tasks.

In analyzing the performance of our Algorithm, we noticed that the Hungarian method (for solving LAPs) occupies about 46% of the total runtime. The optimization of the Hungarian method would allow us to significantly reduce the runtime. As the algorithm is written now, the LAPs are always solved starting with the original costs, even though one LAP differs from another by a single column of costs. If we can take into account information from solving one LAP to aid in solving another, much processing time can be saved.

Another important point to be tackled is the fault tolerance of parallel algorithms executed on cluster or grid. This is important mainly due to our long execution time applications. This problem is included in the version 3 of Athapascan.

References

[1] W.P. Adams et H.D. Sherali. A Tight linearization and an algorithm for Zero-One Quadratic Programming Problems, Management Sciences 32:1274-1290,1986.

[2] W.P. Adams et T.A. Johnson. Improved Linear Programming-Based Lower Bounds for the Quadratic Assignment Problem, DIMACS Series in Discrete Mathematics and Theoritical Computer Science 16:43-76,1994.

[3] K.M. Anstreicher et N.W. Brixius. A New Bound for the Quadratic Assignment Problem Based on Convex Quadratic Programming. Mathematical Programming 89:341-357,2001.

[4] K.M. Anstreicher, N.W. Brixius, J-P Goux et J. Linderoth. Solving large quadratic assignment problems on computational grids, to appear in Mathematical Programming, Series B, Currently available on the Web from http://www.biz.uiowa.edu/faculty/anstreicher.mwqap.ps.2001.

[5] J.L. Roch, T. Gautier et R. Revire ATHAPASCAN: an API for Asynchronous Parellel Programming. User's Guide Rapport technique, ID-IMAGronoble, N°0276, Fevrier2003.

[6] R.E. Burkard. Locations with Spatial Interactions: the Quadratic Assignment Problem. In P.B. Mirchandani and R.L. Francis, editors, Discrete Location Theory, P.B. Mirchandani and R.L. Francis, eds., John Wiley et Sons:387-437,1990.

[7] P.M. Hahn et T. Grant. Lower bounds for the quadratic assignment problem based upon a dual formulation. Operations Research 46:912-922,1998.

[8] P.M. Hahn et J. Krarup. A hospital facility layout problem finally solved. The Journal of Intelligent Manufacturing, vol. 12, N° 5/6:487-496, 2001. Also on web site: http://www.seas.upenn.edu/ hahn/

[9] T. C. Koopmans et M. J. Beckmann. Assignment problems and the location of economic activities. Econometrica 2:53-76,1957.

[10] P.M. Pardalos, F. Rendel et H. Wolcowisz. The quadratic assignment problem: A survey and recent developments. DIMACS Series Discrete Mathematics and Theorical Computer Science 16:1-42,1994.

[11] H.D. Sherali et W.P. Adams. A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems, SIAM Journal on Discrete Mathematics, vol. 3, N°3:411-430, 1990.