Towards a Napster-Like P2P B&B algorithm

M. Mehdi, M. Mezmaz, N. Melab and E-G. Talbi
Laboratoire d'Informatique Fondamentale de Lille
UMR CNRS 8022, INRIA Futurs - DOLPHIN Project
Cité scientifique - 59655, Villeneuve d'Ascq cedex - France
{mehdi,mezmaz,melab,talbi}@lifl.fr

Abstract— **The Branch and Bound (B&B) algorithm is one of the most used methods to solve in an exact way combinatorial optimization problems. In a previous article, we proposed a new approach of the parallel B&B algorithm for distributed systems. This approach is based on a new way to efficiently deal with some crucial issues met in distributed systems. The new method is used to propose a parallilization of the B&B with the farmer-worker paradigm. The obtained resluts show the efficiency and the scalability of the approach.**

However, the new farmer-worker approach has a disadvantage: some nodes of the B&B tree can be explored by several B&B processes. To avoid this redundant work, we propose a new approach based on the Napster-like Peer-to-Peer(P2P) model. Validation is performed by experimenting the approach on a bi-objective flow-shop problem instance that has never been solved exactly. The obtained results, after 15 days on computation pool of about 2500 processors, belonging to 8 distinct clusters, prove the efficiency of the proposed approach. Indeed, the peer processors were exploited on average to 99.3% while the index processor was exploited 0.01%.

Keywords— **Branch and Bound, Parallel Computing, Peer-to-Peer Computing, Flow-Shop Problem.**

I. Introduction

Combinatorial optimization addresses problems for which the resolution consists in finding the (near-)optimal configuration(s) among a large finite set of possible configurations. In practice, most of these problems are naturally NP-hard and complex. The Branch and Bound (B&B) algorithm is one of the most popular methods to solve exactly this kind of problems. This algorithm allows to reduce considerably the computation time required to explore all the solution space associated with the problem being solved. However, the exploration time remains considerable, and using parallel processing is one of the major and popular ways to reduce it. Many parallel B&B approaches have been proposed in the literature. A taxonomy of associated parallel models is presented in [12]. Four models are mainly identified and studied within the context of distributed computing. The parallel exploration of the search tree is one the most used one.

[13] proposes a farmer-worker approach based on special coding of the explored tree and the work units. These codings allow to optimize the dynamic distribution and check-pointing mechanisms on distributed systems, to implicitly detect the termination of the algorithm, and to efficiently share the global information. The algorithm has been applied to the flow-shop scheduling problem, one of the hardest challenging problems in combinatorial optimization. Using the new approach, the problem instance (50 jobs on 20 machines) has been optimally solved for the first time. The method allows not only to improve the best known solution for the problem instance but it also provides a proof of the optimality of the provided solution.

However, the new farmer-worker approach has a disadvantage: some nodes of the tree can be explored by several B&B processes. To avoid this redundant work, it is indispensable for the B&B processes to communicate and cooperate during the resolution. The approach thus must be deployed according to the (Peer-to-Peer)P2P paradigm. The strategies of Napster [9] and Gnutella [11] are the two main approaches used in P2P systems. Unlike the approach of Gnutella, the model of Napster requires minor modifications to adapt the farmer-worker paradigm and to get a P2P deployment without a redundancy in the tree exploration. To the best of our knowledge, the presented approach in this paper is the first use of the Napster-like P2P model for computing systems. The goal of the new approach is to be as efficient as the farmer-worker approach, to avoid the redundancy in work during the exploration of the B&B tree, and to prove that Napster-like P2P model is efficient for computing systems. Unlike the farmer-worker approach, the Napster-like P2P approach can be compared easily with the sequential B&B since no node of the tree is visited more than one time.

The rest of the paper is organized as follows. **Section II** and **Section III** give an overview of the B&B algorithm, its parallelization, and related works. The parallel approach proposed in [13] is described in **Section III**. **Section V** presents its implementation based on the farmer-worker paradigm. In **Section VI**, we describe our new Napster-like P2P approach for the parallelization of the B&B algorithm. **Section VII** presents the experiment performed on a bi-objective flow-shop instance to evaluate the quality of the approach. **Section VIII** draws some conclusions and perspectives of this work.

II. Parallel B&B Algorithm

Solving a problem in combinatorial optimization consists in exploring a search space to provide a (near-)optimal solution. To each candidate solution of the

This work is part of the *CHallenge in Combinatorial Optimization (CHOC)* project supported by the *National French Research Agency (ANR)* through the *Hign-Performance Computing and Computational Grids (CIGC)* programme.

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

search space is associated a cost. Solving exactly a combinatorial optimization problem consists in finding the solution having the optimal cost. For this purpose, the B&B algorithm is based on an implicit enumeration of all the solutions of the considered problem. The search space is explored by dynamically building a tree whose root node represents the problem being solved and its whole associated search space. The leaf nodes are the potential solutions and the internal nodes are subspaces of the total solution space. The size of these subspaces is increasingly reduced as one approaches the leaves.

The construction of such a tree and its exploration are performed using four operators: *branching*, *bounding*, *selection* and *elimination*. The algorithm proceeds in several iterations during which the best solution found so far is progressively improved. The generated and not yet treated nodes are kept in a list whose initial content is limited to only the root node. The four operators intervene at each iteration of the algorithm. The B&B makes it possible to reduce considerably the computation time necessary to explore the whole solution space. However, this remains considerable and parallel processing is thus required to reduce the exploration time.

In [12], four parallel models are identified for B&B algorithms: (1) *the parallel multi-parametric model*, (2) *the parallel tree exploration*, (3) *the parallel evaluation of the bounds*, and (4) *the parallel evaluation of a single bound*. The model (1) consists in launching simultaneously several B&B processes. These processes differ by one or more operators, or have the same operators, but parameterized differently. The trees explored in this model are not necessarily the same. Model (1) guarantees the implicit exploration of the whole solution space. Like model (1), model (2) also consists in launching several B&B processes. However, all the processes in model (2) are similar, and explore simultaneously the same tree. Among the four models, this model is the most popular and studied one. Unlike the two previous models, models (3) and (4) suppose the launching of only one B&B process. They do not allow to parallelize the whole B&B algorithm as both models (1) and (2) do, but they parallelize only the bounding operator. In model (3), each process evaluates the bounds of a distinct pool of nodes, while in model (4) a set of processes evaluate in parallel the bound of a single node.

In [12], an analysis of these different parallel models is presented within the context of distributed computing systems. Distributed systems in general, and P2P systems in particular, exploit the resources of a great number of machines. These resources can be processors, memories or others. A distributed system aims at giving the illusion of a very powerful virtual machine. It makes it possible to solve problems which require very long execution time. Since distributed systems are distributed memory systems, the parallel tree exploration model is more suitable for these environments.

III. Related Works

Many parallel B&Bs on distributed computing systems are described in the literature. [12], [5] and [15] present different parallel strategies for the B&B algorithm. On the distributed systems, the B&B algorithm is often deployed according to the master-slave paradigm. The master manages the list of the not yet explored nodes and distributes nodes to the slave machines. A slave machine receives one node only from the master, explores the subtree of which the received node is the root, and returns to the master all not explored nodes. From a deployment to another, what change often is the condition of returning these nodes. In [3], a slave machine returns all the not explored nodes after a hundred seconds. In [1], a slave machine explores only the son nodes of the received node, and returns the result to the master.
The best parallel efficiency recorded by [3], on a platform of 185 processors, is equal to 85.6%. This result is obtained by exploiting on average only 17 processors during this test. While the best parallel efficiency recorded by [1], on a grid of 128 processors, is equal to 71%. Besides, [2] shows the limits of this paradigm and the used load balancing strategy. [2] advises to use to the hierarchical master-slave paradigm. However, by using 348 processors organized with hierarchical master-slave paradigm, the best parallel efficiency obtained by [2] is about 33%.
[8] proposes an original P2P strategy. These strategy is often referenced in the combinatorial optimization literature on distributed systems. However, the obtained parallel efficiency are less of the one obtained in [3]. Indeed, [8] obtains 84.4% on a simulator of 100 processors.

IV. Fold-Unfold Approach

The proposed approach in [13] is based on the *parallel tree exploration model* with a depth first *search strategy*. This approach is focused on the list of active nodes. The B&B active nodes are those generated but not yet treated. During a resolution, this list evolves constantly and the algorithm stops once it becomes empty. Any list of active nodes covers a set of tree nodes. This set is made up by all nodes which can be explored from a node of this active list. The principle of the approach is based on the assignment of a number to each node of the tree. The numbers of any set of nodes, covered by a list of active nodes, always form an interval. The approach thus defines a relation of equivalence between the concept of *list of active nodes* and the concept of *interval*. So, it is possible to deduce a list of active nodes from an interval, and an interval from a list of active nodes. As its size is reduced, the interval is used for communications and check-pointing, while the list of active nodes is used for exploration.
In order to pass from one concept to the other, the approach defines two additional operators: the *fold opera-*

tor and the *unfold operator*. The fold operator deduces an interval from a list of active nodes, and the unfold operator deduces a list of active nodes from an interval.

V. Farmer-Worker Parallel Approach

Fold and Unfold operators can be used for the parallelization of the B&B according to different parallel paradigms. In [13], the selected paradigm is the farmer-worker one. In this paradigm, only one host plays the role of the farmer, and all the other hosts play the role of a worker. This paradigm is relatively simple to be used. Its major disadvantage is that the farmer can constitute a bottleneck. However, communicating and handling intervals instead of list of active nodes make it possible to reduce the communication costs and the farmer work. This paradigm is thus selected in [13] to test the fold-unfold approach. The goal is to show that the approach makes it possible to push the limit of this paradigm as for the bottleneck, and to have thus a more scalable approach.

In the adopted farmer-worker approach, the workers host as many B&B processes as they have processors, and the farmer hosts the coordinator. Each B&B process explores an interval of node numbers, and manages local solution set. To get a work, a B&B process obtains an interval from the coordinator, deduces a list of nodes from this interval using the unfold operator, and explores this node list. To do a check-point, a B&B process deduces an interval from the list of not treated nodes using the Fold operator, and communicates this interval to the coordinator. The cost of the fold and unfold operators are infinitely negligible compared to the time devoted to exploring the B&B tree. On the other hand, the coordinator keeps a copy of all the not yet explored intervals, and manages global solution set. The copies of the intervals are kept in a set noted $INTERVALS$, and the solutions of the global solutions in another set noted $SOLUTIONS$. Fig. 1 gives an example with three B&B processes and a coordinator. In this example, three intervals are being explored, and the fourth one is waiting for a free available B&B process.

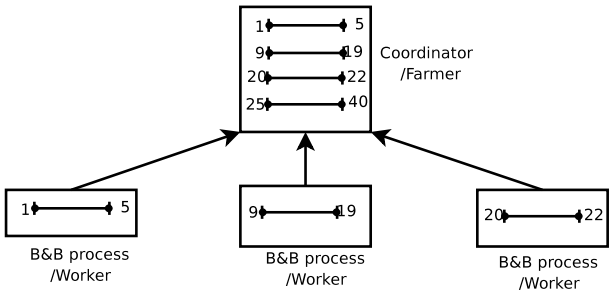

Fig. 1. An example with B&B processes and a coordinator

In addition to balancing the load between B&B processes, other problems must be taken into account. Indeed, the B&B processes make two assumptions about the workers. They suppose that they are likely to break down and not necessary dedicated. Consequently, these processes are fault tolerant and are launched according to the cycle stealing model. The only assumption of the coordinator about the farmer is that it can fail. The coordinator manages only the possible failures of the farmer.

The approach presented in [13] has been implemented following a large scale idle time stealing paradigm (Farmer-Worker). It has been experimented on a flow-shop problem instance ($Ta056$) that has never been optimally solved. The new algorithm allowed to realize a success story as the optimal solution has been found with proof of optimality, within 25 days using about 1900 processors belonging to 9 Nation-wide distinct clusters (administration domains). During the resolution, the worker processors were exploited with an average of 97% while the farmer processor was exploited only 1.7% of the time. These two rates are good indicators on the efficiency of this approach and its scalability.

VI. Napster-Like P2P Approach

However, the presented farmer-worker approach in [13] has a disadvantage: some nodes of the tree can be explored by several B&B processes. Let [A,B[an interval being explored by a holder B&B process, and [A',B'[its copy in the coordinator. As explained in [13], the interval [A',B'[can be divided into two intervals [A',C[and [C,B'[. This occurs after a request from requesting B&B process. After this division, [C,B'[is explored by the requesting process, while the holder process continues the exploration of [A,B[. However, [C,B'[and [A,B[are not completely disjoint. Consequently, the same nodes of the tree can be explored by the two processes.

To avoid this redundant treatment, it is indispensable for requesting and holder processes to coordinate their intervals. Before beginning the exploration of the received interval, the requesting process must make sure that the two intervals are disjoint. It is essential to requesting process to contact the process holder in order to refine the division. The holder process is then given the responsibility to determine the intervals that each of the two processes must explore. This is done by the intersection and subtraction operators. These two operators are noted $\cap$ and $\setminus$, respectively. The equations (1) and (2) define them.

$$[A,B[\cap[C,D[\;=\; [max(A,C),min(B,D)[\tag{1}$$

$$[A,B[\setminus[C,D[\;=\; \{X/X \in [A,B[\;\; and \;\; X \notin [C,D[\} \tag{2}$$

Let [A,B[the interval of the holder process, and [C,D[the received interval by the requesting process. The holder process proceeds in two stages. First, it subtracts from [C,D[the already explored interval by the

holder process. Then, it subtracts [A,B[from the given interval to the requesting process. In other words, $[C, D[$ and $[A, B[$ become equal to $[C, D[\cap[A, B[$ and $[A, B[\backslash[C, D[$, respectively. Once this new division finished, both intervals obtained are completely disjoint, and both processes can continue the exploration of their interval. This new strategy thus ensures that no nodes is explored twice.

Unlike the previous approach, this new approach supposes that a B&B process can be contacted by another B&B process. Indeed, the B&B processes send requests to the coordinator and to the other B&Bs processes, and can receive requests from any B&B process. The approach thus must be deployed according to the P2P paradigm. In this new approach, a peer hosts as many B&B processes it has processors. The intervals constitute the handled resources. In a P2P deployment, one important issue must be taken into account: the resource discovery and their routing.

The resource discovery consists in identifying the peer able to provide the required resource. As indicated in [4], the strategies of Napster [9] and Gnutella [11] are the two main approaches used in P2P systems.

In Naspter, the resource discovery is centralized, and based on an index peer which stores the peer-resource relations. When a peer seeks a resource, it obtains from the index the address of the peer having this resource.

Unlike the centralized strategy of Napster, the resource discovery in Gnutella is completely distributed. In Gnutella, a peer is connected only by its neighbors. They are logical neighbors and the set of connections forms logical topologies. To find a resource, a peer broadcast a message to its neighbors in this logical topology. The request is propagated by a neighbor to its own neighbors. A peer, which receives the request and which has the required resource, does not propagate the message, and returns the resource to the requesting peer.

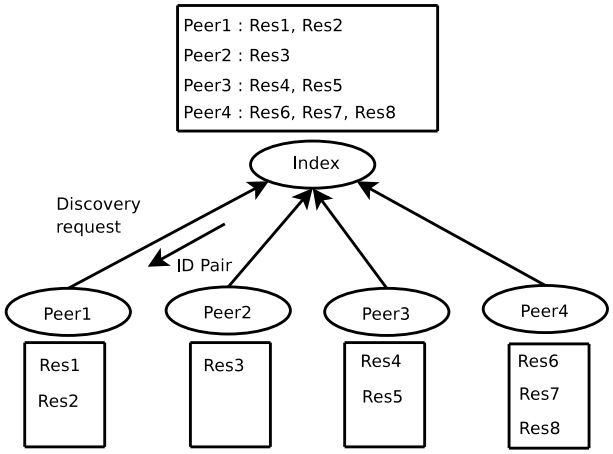

Fig. 2. Resource discovery in Napster

To deploy the B&B, our approach is based on the resource discovery model of Napster. Unlike the approach of Gnutella, the model of Napster requires minor modifications to adapt the farmer-worker paradigm, and to get a P2P deployment without a redundancy in the tree exploration. The coordinator process plays the role of index, and the worker processes the role of the peers. In the farmer-worker paradigm, the coordinator returns to a worker an interval. In this new strategy, the index returns to a peer the resource and the address of the peer which holds this resource. As already explained, this resource is an interval. Then, the requesting peer contacts the holder peer of the received interval in order to make disjoint the intervals of the holder and requesting peers.

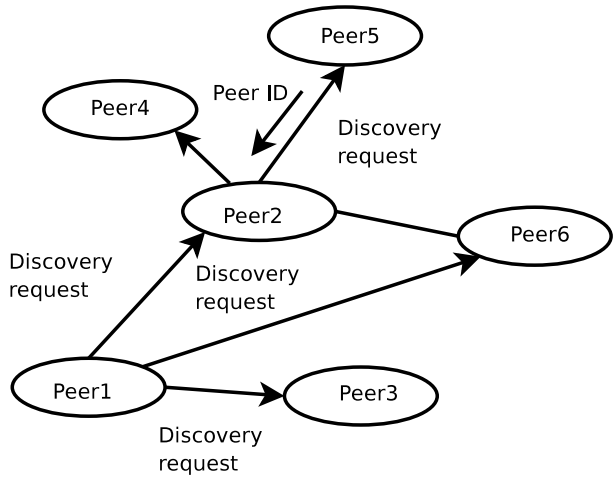

Fig. 3. Resource discovery in Gnutella

VII. Experimentation

A. Bi-objective permutation flow-Shop problem

The flow-shop problem is one of the numerous scheduling multi-objective problems [14] that has received a great attention given its importance in many industrial areas. The problem can be formulated as a set of N jobs $J_1, J_2, \ldots, J_N$ to be scheduled on M machines. The machines are critical resources as each machine can not be simultaneously assigned to two jobs. Each job J_i is composed of M consecutive tasks $t_{i1}, \ldots, t_{iM}$, where t_{ij} designates the j^{th} task of the job J_i requiring the machine m_j. To each task t_{ij} is associated a processing time p_{ij}, and each job J_i must be achieved before a due date d_i.

The problem being tackled here is the bi-objective permutation flow-shop problem where jobs must be scheduled in the same order on all the machines. Therefore, two objectives have to be minimized: (1) C_{max}: makespan (Total completion time), (2) T: total tardiness. The task t_{ij} being scheduled at time s_{ij}, the two objectives are NP-hard[7][10], and can be formulated as follows:

$$C_{max} = Max\{s_{iM} + p_{iM} | i \in [1 \ldots N]\}$$

$$T = \sum_{i=1}^{N} [max(0, s_{iM} + p_{iM} - d_i)]$$

The application of the proposed approach to the flow-hop problem has been experimented on one of the instances proposed by [6]. More exactly, it is an instance generated for problems of 50 jobs on 5 machines in

which only the makespan[1] is considered. The instance has been extended with the tardiness[2] as the second objective. Such instance has never been solved exactly in its bi-objective formulation.

B. The experimentation platform

The method is tested on the computational pool detailed in Table I. It is made up of approximately $2,500$ processors belonging to 8 clusters. These clusters belong to Grid'5000[3]. Grid'5000 is a Nation-wide experimental grid composed by 9 clusters distributed over several French universities (Bordeaux, Grenoble, Lille, Nancy, Orsay, Rennes, Sophia, Toulouse). The eight exploited clusters are those of Bordeaux, Lille, Nancy, Orsay, Rennes, Sophia, Toulouse. All the machines of Grid'5000 are dedicated bi-processors, and interconnected by the Ethernet Gigabit, using RENATER[4] nation-wide network.

CPU model	Cluster	Number of CPU
AMD 2.2	Bordeaux	2x58
Xeon 3.0		2x43
AMD 2.2	Lille	2x53
AMD 2.6		2x46
AMD 2.0	Lyon	2x56
AMD 2.4		2x70
AMD 2.0	Nancy	2x47
Xeon 1.6		2x120
AMD 2.0	Orsay	2x216
AMD 2.4		2x126
AMD 2.2	Rennes	2x64
AMD 2.0		2x100
AMD 2.0		2x74
AMD 2.2	Sophia	2x56
AMD 2.6		2x50
AMD 2.2	Toulouse	2x58
Total		**2,474**

TABLE I: The computational pool

C. Experimental results

After about 15 days of computation, the experiment did not make it possible to solve the instance. However, the recorded statistics seem enough to evaluate the quality of the approach. Table II summarizes the most important statistics recorded during the resolution. The experiment lasted approximately 15 days, with an average of 655 processors, a maximum of $1,606$ available processors, and a cumulative computation time of about 26 years. About 7 billion nodes were explored.

As Table II indicates, more than 4 million check-point operations were done by the B&B processes, while the index did its check-point about 7 hundred times. These check-point operations have allowed the fault tolerance mechanism to face the thousands of failures which have occurred. Indeed, more than 150 thousand failures of the peers and 31 failures of the index were recorded.

The peer processors were exploited on average to 99.3% while the index processor was exploited 0.01%. These two rates are good indicators on the parallel efficiency of this approach and its scalability. In a Napster-like P2P paradigm, a good approach must maximize the exploitation rate of the peer processors and must minimize the exploitation rate of the index processor.

Running wall clock time	15 days
Total CPU time	26 years
Average number of peers	655
Maximum number of peers	1,606
Number of explored nodes	6,782,787,073
Index check-point operations	698
Peer check-point operations	4,581,950
Peer failures	169,141
Index failures	31
Peer CPU exploitation	99.3%
Index CPU exploitation	0.01%

TABLE II: The computation statistics

VIII. Conclusions and Future Works

Solving exactly large instances of combinatorial optimization problems requires a huge amount of computational resources. Parallel Branch and Bound(B&B) algorithms based on the parallel exploration of the search tree have successfully been applied to solve these problems. However, experiments are often limited to few tens of processors. Designing and implementing B&B algorithms for a large scale computational distributed systems is a great research challenge as several crucial issues must be tackled. In [13], we have proposed a new B&B algorithm with new approaches allowing to efficiently tackle the problems met in distributed systems. The approach consists in an efficient coding associated with the explored tree and work units (collections of nodes). The approach has been implemented following a large scale idle time stealing farmer-worker paradigm. The obtained resluts show the efficiency and the scalability of the approach.

However, the new farmer-worker approach has a disadvantage: some nodes of the B&B tree can be explored by several B&B processes. To avoid this redundant work, we propose a new approach based on the Napster-like Peer-to-Peer(P2P) model. Validation is performed by experimenting the approach on a bi-objective flow-shop problem instance that has never been solved exactly. The obtained results, after 15 days on a computation pool of about 2500 processors, belonging to 8 distinct clusters, prove the efficiency of the proposed approach. Indeed, the peer processors were exploited on average to 99.3% while the index processor was exploited 0.01%. These two rates are good indicators on

[1] http://www.eivd.ch/ina/Collaborateurs/etd/default.htm
[2] http://www.lifl.fr/OPAC/
[3] http://www.grid5000.fr
[4] http://www.renater.fr

the parallel efficiency of this approach and its scalability. To the best of our knowledge, the presented approach is the first use of the Napster-like P2P model for computing systems. The new approach is as efficient as the farmer-worker approach, avoid the redundancy in work during the exploration of the B&B tree, and prove that Napster-like P2P model is efficient for computing systems.

Unlike the farmer-worker approach, the Napster-like P2P approach can be compared easily with the sequential B&B since no node of the tree is visited more than one time. We plan to study the variation of efficiency according to the number of peers. It is also planned to use the approach with an other P2P paradigm to push far the scalability limits of the Napster-like P2P model. The objective is to exploit more and more processors and to solve more and more complex instances.

Acknowledgment

We would like to thank the *National French Research Agency* (ANR). This work is part of the *CHallenge in Combinatorial Optimization (CHOC)* project supported by the ANR through the *High-Performance Computing and Computational Grids (CIGC)* program. Our thanks are also addressed to the persons in charge, to the engineers and to the technicians of the Grid'5000 clusters.

References

[1] K. Aida and Y. Futakata. High-performance parallel and distributed computing for the BMIeigenvalue problem. *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 71–78, 2002.

[2] K. Aida and T. Osumi. A case study in running a parallel branch and bound application on the grid. *Applications and the Internet, 2005. Proceedings. The 2005 Symposium on*, pages 164–173.

[3] K. Anstreicher, N. Brixius, J.P. Goux, and J. Linderoth. Solving large quadratic assignment problems on computational grids. *Mathematical Programming*, 91(3):563–588, 2002.

[4] Franck Cappello. Calcul Global Pair a Pair : extension des systemes Pair a Pair au calcul. *Lettre de l'IDRIS*, pages 14–25, month=Janvier,, 2002.

[5] D.Gelenter and T.G.Crainic. Parallel Branch and Bound Algorithms: Survey and Synthesis. *Operation Research*, pages 42:1042–1066, 1994.

[6] E.Taillard. Banchmarks for basic scheduling problems. *European Journal of European Research*, pages 23:661–673, 1993.

[7] Johnson D.S. Garey M.R. and Sethi R. The complexity of flow-shop and job-shop scheduling. *Mathematics of Operations Research*, 1:117–129, 1976.

[8] A. Iamnitchi and I. Foster. A Problem-Specific Fault-Tolerance Mechanism for Asynchronous, Distributed Systems. *29th International Conference on Parallel Processing (ICPP), Toronto, Canada, August*, pages 21–24, 2000.

[9] N. Inc. The Napster homepage. *Online: http://www. napster. com*, 2000.

[10] Du J. and Leung J. Y.-T. Minimizing Total Tardiness on One Machine is NP-hard. *Mathematics of operations research*, 15:483–495, 1990.

[11] G. Kan. Gnutella. *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*, 2001.

[12] N. Melab. *Contributions à la résolution de problèmes d'optimisation combinatoire sur grilles de calcul.* PhD thesis, LIFL, USTL, Novembre 2005.

[13] M. Mezmaz, N. Melab, and E-G. Talbi. A Grid-enabled Branch and Bound Algorithm for Solving Challenging Combinatorial Optimization Problems. In *In Proc. of 21th IEEE Intl. Parallel and Distributed Processing Symp.*, Long Beach, California, March 2007.

[14] V. T'kindt and J-C. Billaut. *Multicriteria Scheduling - Theory, Models and Algorithms.* Springer-Verlag, 2002.

[15] H. Trienekens and A. de Bruin. Towards a taxonomy of parallel branch and bound algorithms. Report EUR-CS-92-01, Department of Computer Science,Erasmus University Rotterdam, 1992.