A Parallel Exact Hybrid Approach for Solving Multi-Objective Problems on the Computational Grid*

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
{mezmaz,melab,talbi}@lifl.fr

Abstract

This paper presents a parallel hybrid exact multi-objective approach which combines two metaheuristics - a genetic algorithm (GA) and a memetic algorithm (MA), with an exact method - a branch and bound (B&B) algorithm. Such approach profits from both the exploration power of the GA, the intensification capability of the MA and the ability of the B&B to provide optimal solutions with proof of optimality. To fully exploit the resources of a computational grid, the hybrid method is parallelized according to three well-known parallel models - the island model for the GA, the multi-start model for the MA and the parallel tree exploration model for the B&B. The obtained method has been experimented and validated on a bi-objective flow-shop scheduling problem. The approach allowed to solve exactly for the first time an instance of the problem - 50 jobs on 5 machines. More than 400 processors belonging to 4 administrative domains have contributed to the resolution process during more than 6 days.

Keywords: *Multi-Objective Optimization ; Hybridization; Parallel Computing; Genetic/Memetic Algorithm; Branch and Bound; Flow-Shop.*

1. Introduction

Combinatorial optimization addresses problems for which the resolution consists in finding the optimal configuration(s) among a large finite set of possible configurations. Most of these problems are NP-hard and multi-objective in practice. Their resolution is often performed either with exact or near-optimal methods. The best results have often been provided by hybrid approaches combining both of these different methods [8]. Nevertheless, as the hybridization mechanism is CPU time-consuming it is not often fully exploited in practice. Indeed, experiments with hybrid algorithms are often stopped before the convergence is reached [1]. Solving large size and time-intensive combinatorial optimization problems with parallel hybrid optimization algorithms requires a large amount of computational resources. Grid computing is recently revealed as a powerful way to harness these resources and efficiently deal with such problems.

In this paper, we are interested a parallel hybrid approach that combines a GA and an MA (including a local search method) to provide the AGMA algorithm [1]. This latter is powerful as it profits from the exploration power of the GA and the intensification capability of the MA. AGMA is then combined with a B&B algorithm to provide a hybrid method which is able to produce efficiently exact solutions to multi-objective problems. Different models have been proposed in the literature for the parallel design and implementation of optimization methods [6]. Three of them are exploited in this paper: the island model, the multi-start model and the parallel exploration of search tree. The island model allows to provide more effective, diversified and robust solutions by delaying the global convergence of the GA. The multi-start model allows the parallelization of the local search phase of the MA algorithm. The parallel exploration of search tree allows to speed up the execution of the B&B algorithm.

The proposed approach has been experimented on the bi-objective permutation flow-shop problem [10]. The problem consists roughly to find a schedule of a set of jobs on a set of machines that minimizes the makespan and the total tardiness. Jobs must be scheduled in

*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.

1-4244-0054-6/06/$20.00 ©2006 IEEE

the same order on all machines, and each machine can not be simultaneously assigned to two jobs. The parallel hybrid approach has been applied to such problem. The approach allowed to solve exactly for the first time an instance of the problem - 50 jobs on 5 machines. More than 400 processors belonging to 4 administrative domains have contributed to the resolution process during more than 6 days.
The rest of this paper is organized as follows: Section 2 highlights the major features of multi-objective optimization and presents an overview of GAs, MAs and B&B. Sections 3 and 4 describe the hybridization and parallelization of the combined algorithms respectively. Section 5 formulates the flow-shop permutation problem and reports the obtained experimental results. The conclusion is drawn in Section 6.

2. Multi-objective combinatorial optimization

2.1. Concepts and definitions

In combinatorial optimization, a problem can be mono-objective or multi-objective whether one is interested respectively in one or more than one objective. A cost is associated to each solution of the problem being tackled, and this cost can be according to the number of objectives, either a simple value or a vector of values. Solving a combinatorial optimization problem consists in finding the solution(s) having the optimal cost. If the optimality concept is simple to define in the case of mono-objective problems, it is not obvious for multi-objective problems. Optimality for multi-objective problems is generally defined using the relation of *dominance* between vectors.

Definition.1:
Let $X = (X_1, .., X_N)$ *and* $Y = (Y_1, .., Y_N)$ *be two vectors and i,j* $\in$ *[1,N] two integers. The dominance relation is defined as follows:*
X dominates $Y \Longleftrightarrow (\forall i, X_i \leq Y_i)$ *and* $(\exists j, X_j < Y_j)$

The dominance relation constitutes a partial order, implying that several optimal solutions may exist. The set of optimal solutions is called *Pareto Optimal Set*, and the *Pareto front* is the corresponding set of the Pareto Optimal Set in the objective space.

Definition.2:
Let $X = (X_1, .., X_N)$ *and* $Y = (Y_1, .., Y_N)$ *be two vectors and i,j* $\in$ *[1,N] two integers. Let E be a set of vectors, and F the Pareto front of E,*

$X \in F \Longleftrightarrow \nexists\ Y \in E\ /\ Y$ *dominates* X.

2.2. Resolution methods

In practice, there is a broad range of NP-hard discrete multi-objective optimization problems (MOPs). Basically, two major approaches are often used to tackle these problems: exact methods and metaheuristics. Exact methods allow to find exact solutions but they are impractical as they are extremely time-consuming. Conversely, the use of metaheuristics generally meets the needs of decision makers to efficiently generate "satisfactory" solutions. In this work, we are interested in two metaheuristics GA and MA and one exact method i.e. B&B.
- GAs are population-based metaheuristics based on the iterative application of stochastic operators on a population of candidate solutions.
- MAs have strong similarities with "classical" GAs. They are designed in order to speed up the convergence of GAs, considered to be slower. The principal idea is to include a local search mechanism in a GA process by replacing one of its genetic operators. For this reason, MAs are often considered as GAs hybridized with a local search.
- B&B algorithms are based on an implicit enumeration of all the solutions of the considered problem. The solution 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 possible solutions and the internal nodes are subspaces of the total solution space.

3. A Multi-objective exact hybrid approach

In order to take advantage of the benefits brought by various methods, it is often necessary to combine them. Nowadays, hybrid methods allow to obtain the best results on the majority of the academic and practical problems. In our work, we addressed the high level hybridization with the co-evolutionary and relay modes. In the high level hybridization, the internal structure of a method is not modified unlike in the low level, where a resolution method is inserted into another one. In the relay mode, the methods are sequentially executed contrary to the co-evolutionary mode where they are simultaneously executed. A complete presentation of the various modes and levels of hybridization can be found in [8].
In this paper, we are interested in combining the three optimization methods presented above: GA, MA and

B&B. The objective is to take benefits from each of them: exploration powerful of the GA, search intensification capability of MA and the ability of B&B to provide exact solutions.

3.1. AGMA: A hybridization of GA & MA

In single objective optimization, it is well known that GAs provide better results when they are hybridized with local search algorithms. Indeed, the GA convergence is too slow to be really effective without any cooperation. In [1], a hybrid genetic-memetic algorithm named *AGMA* that combines GA and an MA has been proposed. In this paper, we do not give the details and parameters of the two algorithms, and if needs be, the reader is referred to [1]. The GA uses mainly two parameters: an archive (Pareto front) PO^* of non-dominated solutions, and a progression ratio P_{PO^*} of PO^*. At each generation, these two parameters are updated. If no significant progression is noticed ($P_{PO^*} < \alpha$, where α is a fixed threshold), an intensified search process is triggered. The intensification consists in applying MA to the current population during one generation. The application of MA returns a Pareto front $PO^{*'}$ that serves to update the Pareto front PO^* of the GA.
MA consists in selecting randomly a set of solutions from the current population of the GA. A crossover operator is then applied to these solutions and new solutions are generated. Among these new solutions only non-dominated ones are maintained to constitute a new Pareto front $PO^{*'}$. A local search is then applied to each solution of $PO^{*'}$ to compute its neighborhood. The non-dominated solutions belonging to the neighborhood are inserted into $PO^{*'}$.

3.2. Hybridization of AGMA with B&B

The goal of the hybridization with B&B is to exploit the complementary advantages of both AGMA and B&B. Indeed, the AGMA algorithm allows to provide efficiently near-optimal solutions. The B&B algorithm exploits these solutions as lower bounds to eliminate a large number of nodes, and thus to provide more efficiently exact solutions. In this work, we exploited and experimented two high-level hybridization modes: relay and co-evolutionary. In the relay mode, B&B is initialized with the Pareto front provided by AGMA. In the co-evolutionary mode, B&B and AGMA are deployed simultaneously and cooperate by exchanging solutions of their Pareto fronts. In both modes, the role of AGMA is to provide B&B with good solutions in order to eliminate earlier B&B nodes that hold bad solutions.

4. A Multi-level parallelization of the approach

Nowadays, parallel computing is more and more performed on computational grids. These systems exploit resources (processors, memory, etc.) of thousands of computers offering the illusion of an extremely powerful virtual unique computer. They make it possible to solve problem instances which require a very long execution time. A computational grid represents a virtual infrastructure built of a coordinated shared set of computational resources, distributed and heterogeneous, for which there is no centralized administration. Many middlewares are proposed to facilitate their exploitation. However, they do not often allow to support parallel computing. Indeed, many of them do not provide tools to make possible the cooperation between parallel tasks. Therefore, we have proposed in [7] a coordination model as an extension of these middlewares. The model is particularly adapted to multi-objective combinatorial optimization.

4.1. The coordination model

Several coordination models have been proposed in the literature, and Linda [3] is certainly the most popular of them. It is based on the generative communication paradigm. In this latter, the exchange of messages between the processes is done via a shared memory in which a sender process puts its message and from which a receiver process recovers it. In Linda, the shared memory and a message are respectively called *tuple space* and *tuple*. The tuple space is a collection of tuples, not a set, given the fact that it can contain several copies of the same tuple. A tuple is a finite ordered series of typed fields, each field containing either a typed value or a process call, named *process field*. A tuple which contains only typed values is called *data tuple*. A tuple which contains at least a process field is called *process tuple*.
As soon as a process tuple finishes its execution, it transforms itself into a data tuple by replacing the process fields by the values returned by their respective process calls. Unlike a data tuple, which is a passive entity, a process tuple is an active entity which exchanges tuples by generating, reading and consuming other tuples. Linda is not completely adequate for parallel multi-objective combinatorial optimization on grids. Therefore, the extensions proposed in [7] consist in adding group, non-blocking and rewriting operations. Group operations are necessary because sets of

solutions (Pareto fronts) are managed. Non-blocking and rewriting operations are important in a volatile and large scale grid.

4.2. Multi-level parallelization

Lot of work was carried out on the parallelization of the combinatorial optimization methods. From the various adopted parallel approaches, a certain number of models are identified [9, 6, 2]. In our work, three models are exploited - the island model for the GA part of AGMA, the multi-start model for the local search part of MA, and the parallel tree exploration model for the B&B algorithm.

4.3. The island model

The island model is inspired by behaviors observed in the ecological niches. In this model, several evolutionary algorithms are deployed to evolve simultaneously various populations of solutions, often called islands. The islands are not independent since solutions are exchanged between them. This exchange aims at delaying the convergence of the evolutionary process and to explore more zones in the solution space. For each island, a migration operator intervenes at the end of each generation. Its role, in particular, consists to decide the appropriateness of operating a migration, to select the population sender of immigrants or the receiver of emigrants, to choose the emigrating solutions and to integrate the immigrant ones. The implementation of the island model using our proposed coordination model for computational grids is based on three types of tuples: *island tuples*, *migration tuples*, and *fault-tolerance tuples*.

- Island tuples: At the beginning, a main program puts in the tuple space as many island tuples as islands to be deployed. An island tuple is a tuple process made up of only one field corresponding to an AGMA. An island tuple contains mainly two parameters which are the island number (or AGMA) and the total number of the deployed islands (or AGMAs). The island numbers are distinct values between 1 and the total number of islands. Once put in the tuple space, the island tuples are deployed by the middleware as AGMAs. In addition to the island tuple, two others tuples are assigned to each island - migration and fault-tolerance tuples. Both are data tuples and are used respectively for migration in the island model and for the fault-tolerance mechanism.

- Migration tuples: A migration tuple has the form $[N, MIGRANTS]$, where N is the number of a given island and $MIGRANTS$ contains its migrant solutions. This kind of tuples is used for the importation and exportation of migrants between the islands. The exportation is done in two stages. First, Pareto front solutions to be exported are selected, then they are put in the migration tuple associated with the island. The importation is done according to a migration topology, and the island whose solutions will be imported is selected.

- Fault-tolerance tuples: Fault-tolerance can be dealt with either at the application or middleware level. In our approach, both levels are exploited. At the middleware level, the adopted strategy consists in re-starting from scratch, with the same parameters, on another machine any broken down process tuple. At the application level, the fault-tolerance is ensured using the fault-tolerance tuples. Only one fault-tolerance tuple is assigned for each island having the following structure $[N, GENERATION, POPULATION, PARETO]$. The four fields designate respectively the island number, the number of its current generation, its current population and its Pareto front. The islands save regularly their state by updating the fields of the their associated fault-tolerance tuple. When an island tuple is launched, its first operation is an attempt to read its fault-tolerance tuples. The existence of this tuple means that the same island number was carried out before, and thus the deployed island is a broken down island restarted by the middleware. In this case the generation number, the population, and Pareto front of the island are updated according to values of the fault-tolerance tuple. Otherwise, the island is deployed with its initial values.

4.4. The multi-start model

The multi-start model consists in simultaneously launching several tasks and gathering their results. This model was exploited for the local search parallelization. A local search consists in generating new solutions from the solutions of the MA, to simultaneously explore the neighborhood of these initial solutions, to merge the neighborhood solutions obtained with the initial solution, to keep only the optimal Pareto solutions, to again explore the neighborhood of the kept Pareto solutions and so on. A local search stops when the neighborhood solutions do not improve the initial solutions. A local search is thus a launching of a series of task sets where each task is an exploration of the neighborhood. The deployment of each task set is done according to the multi-start model.

Only one type of tuple, called *exploration tuple*, is used

for the implementation of this model. They are process tuples which contain two fields - a local search number and a call to the exploration program. This program receives, as arguments, the solutions for which the neighborhoods are visited, and returns back the neighboring solutions. Given the relatively short duration of a neighborhood exploration, no fault-tolerance mechanism is elaborated at the application level. In the case of a machine fault during an exploration process, the middleware ensures its redeployment on another machine with the same parameters.

4.5. The parallel tree exploration model

The parallel tree exploration model consists in visiting in parallel different nodes of the sub-trees defining solution subspaces. It means that the branching, selection, bounding and elimination operators are carried out in parallel by different processes exploring these subspaces. In the majority of the B&B parallelization approaches, the work unit is a list of nodes. Either for load balancing, fault-tolerance, scalability, granularity management or termination detection, exchanging lists of nodes on a computational grid is costly in terms of communication and storage. In order to overcome such limit, we propose another approach to describe work units in B&B that minimizes communication and storage costs involved mainly in work distribution and fault tolerance.

The approach assigns according to depth first strategy a number and a weight to each node of the basic tree associated to the problem being solved. The basic tree is the tree generated without applying any elimination operator. The number associated to each internal node is identical to the number of its left-most child node. Leaf nodes are numbered from the left to right. The root number is thus equal to the number of the left-most leaf node of the basic tree. The weight of a node is the number of the leaves of the sub-tree of which it is the root. The leaf nodes have a weight equal to 1. In the majority of problems, the permutation flow-shop for example, nodes of the same level, even belonging to different sub-trees, have an equal weight. In this approach, a work unit is described by an interval $[x, y[$ indicating all nodes whose numbers are higher than x and strictly lower than y.

Three types of tuples are used for the deployment of the B&B according to this approach: *B&B tuples*, *work tuples* and *solution tuples*.

- B&B tuples: Unlike both other tuples which are data tuples, B&B tuples are process tuples. The deployment of the algorithm is done by deposing as many B&B tuples as B&B processes participating to the computation. As for the island tuples, the middleware is given the responsibility of deploying them on the computational grid.

- Work tuples: Work tuples are associated with different intervals. A work tuple has the form $[N, X, Y]$, where N is the identifier of an interval, X its beginning and Y its end. At the beginning, the tuple space is initialized with only one work tuple covering the totality of the tree nodes. It corresponds to the interval $[1, weight(root)[$. It is given to the first B&B process joining the computation.

When a work tuple $[N_i,X_i,Y_i]$ explored by a process i resumes ($X_i \geq Y_i$) the process i addresses a request to get back work from the tuple space. The tuple space returns back the greatest work tuple not yet allocated, if it exists. Otherwise, the tuple space applies a division operation to the tuple assigned to a process j, ideally corresponding to the biggest interval. Its division results in two tuples $[N_j,X_j,Z]$ and $[N_i,Z,Y_j]$. The process i obtains the latter work tuple and j keeps the former because it already began its exploration from X_j. To avoid the affectation of too fine granularity units, the tuple space uses a threshold below which a tuple is duplicated instead of splitting it. The termination detection is performed in a natural way. Indeed, a tuple $[N_i,X_i,Y_i]$ may be withdrawn if $X_i \geq Y_i$. In this way, the program stops when there are no work tuples in the tuple space.

In addition to the load balancing and termination detection, this approach also facilitates the fault-tolerance management. Periodically, each process sends to the tuple space a report of the progress of its work tuple exploration. If $[N,X_1,Y_1]$ and $[N,X_2,Y_2]$ designate respectively the same work tuple before and during its exploration, the tuple space updates its corresponding interval by applying a tuple fusion reaction which gives the tuple $[N,\text{Max}(X_1,\ X_2),\text{Min}(Y_1,\ Y_2)]$.

- Solution tuples: A solution tuple consists of two fields representing the solution code and its fitness vector. On these tuples, a withdrawal Pareto reaction is defined. A solution tuple is withdrawn from the tuple space if its fitness vector is dominated by the fitness vector of another solution tuple. This withdrawal reaction ensures that only the Pareto solutions are found in the tuple space. Each new Pareto solution found by either a B&B or an island process will be immediately deposited in the tuple space so that the other processes use it. The B&B processes regularly read all the solution tuples to make it possible to the elimination operator to intervene as soon as possible.

Through the use of solution tuples, both hybridization models were designed and implemented in a very simple way. In the relay mode, it is enough to launch the

island processes, to stop them once a stopping criterion (evolution progression) indicates that the Pareto solutions do not improve any more, and to make them followed by the B&B processes described previously. When the island processes resume, the tuple space is not emptied of its Pareto solutions, the B&B processes are thus initialized by the Pareto solutions provided by the island processes. In the co-evolutionary mode, the island and B&B processes are launched at the same time. However, when the island model converges, the island processes are replaced by B&B processes in order to fully exploit the power of the grid. As the two process types share the same tuple space, the solutions found by either island or B&B processes are used by the other ones.

5. Application to the bi-objective flow-shop problem

5.1. Problem formulation

The flow-shop problem is one of the numerous scheduling multi-objective problems [10] 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} represents 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 can be formulated as follows:

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

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

5.2. Experimentation

The application of the proposed parallel hybrid approach to the flow-shop problem has been experimented on one of the instances proposed by [4]. More exactly, it is the second instance generated for problems of 50 jobs on 5 machines in which only the makespan is considered. The instance has been extended with the tardiness as the second objective. Such instance has never been solved exactly in its bi-objective formulation.
The proposed parallel hybrid method presented has been experimented according to various parameters. These parameters concern the three parallel models and the two hybridization types. These parameters and their associated values are the following:
- Hybridization between the GA and the MA: the default parameters associated to the AGMA in [1] are reused.
- Hybridization of the AGMA with the B&B: in either relay or co-evolutionary mode, an island process is stopped if no new Pareto solution is found after 20 minutes. Moreover, in order to fully exploit the computational grid power, a B&B process is deployed in its place.
- The parallel island model: the migration operator and the checkpointing mechanism are triggered in each island every 2 minutes. The exchange of individuals is done according to the random topology. The migrants is the whole Pareto front if it does not contain more than 20 solutions, and only 20 solutions randomly selected from the Pareto front otherwise.
- The multi-start model: each exploration consists in visiting the neighborhood of 11 solutions at the same time.
- The parallel tree exploration: the B&B process contacts the tuple space every 3 minutes in order to save the state of its work and to read the solutions deposited by the other island or B&B processes.

The experimentation material platform is the computational pool detailed in Table 1. It is made up of over 400 machines distributed across four administrative domains belonging to four education departments of the University of Lille1 - the two education (E) and research (R) Gigabit Ethernet domains of Polytech'Lille, the 100 MegaBit Ethernet domain of IUT-A and the Gigabit Ethernet domain of the FIL department. These domains are inter-connected by the Gigabit network of the university. The software grid middleware used for implementation is XtremWeb [5]. This latter is a dispatcher-worker middleware developed at University of Paris Sud. It is basically dedicated to the deployment of multi-parametric applications. We have extended it with the our Linda-like coordination model to deal with parallel cooperative multi-objective optimization [7].

Table 2 summarizes the results obtained with four different experiments. Each experiment corresponds

http://www.eivd.ch/ina/Collaborateurs/etd/default.htm

http://www.lifl.fr/OPAC/

Deployments	Meta. (60 islands)	B&B	Meta.+B&B
Only Meta.	1h43	0	1h43
Only B&B.	0	152h3	152h3
Meta. and B&B in Relay	1h43	116h26	118h9
Meta. and B&B in Cooperation	1h44	128h40	128h40

Table 2. Execution time obtained with and without hybridization

No. of islands	1	10-50	60	70	80	90	100
Time (seconds)	7200	7200	6231	6242	6244	6247	6231

Table 3. Total execution time according to the number of islands

CPU (GHz)	Domain	Role	No.
P4 3.06	Polytech'Lille(R)	Farmer	1
P4 1.70	FIL	Worker	24
P4 2.40			48
P4 2.80			72
P4 3.00			26
AMD 1.30	Polytech'Lille(E)		14
Celeron 2.40			35
Celeron 0.80			14
Celeron 2.00			8
Celeron 2.20			28
P3 1.20			12
P4 3.20			12
P4 1.60	IUT-A		12
P4 2.00			13
P4 2.80			45
P4 2.66			7
P4 3.00			41
Total			412

Table 1. The computational pool

to one raw in the table. The three last columns report respectively the execution time of the two metaheuristics (AGMA), the execution time of the B&B, and their sum.

The first experiment (first raw in Table 2) consists in deploying the AGMA algorithm without B&B. A critical parameter of such deployment is the determination of the convenient number of islands. A trade-off between efficiency and effectiveness has to be found. To do that a series of experiments have been conducted with different values of such parameter. Tables 3 and 4 illustrate respectively the execution times and S-metric values obtained with the different numbers of islands. The S-metric measures the hyper-volume delimited by a reference point and a Pareto front. It allows to evaluate the quality of a Pareto front provided by an algorithm in terms of convergence and diversity. The results show that 60 islands allow to provide efficiently the best Pareto front.

The second experiment (second raw in Table 2) consists in deploying only the B&B processes without any hybridization. As it can be seen in Table 2, the exact Pareto front, plotted in Figure 1, has been found after more than 152 hours (over six days) of computation. The last two experiments (raws 3 and 4 in Table 2) concern the hybridization of AGMA with a B&B in the relay and co-evolutionary modes. The objective of the hybridization is to obtain an optimal solution with proof of optimality with the near-optimal solution provided by the AGMA. In both cases, 60 islands are used as it is the right number of islands to provide efficiently effective solutions. Table 2 shows that the relay mode is faster than the co-evolutionary mode. Indeed, the relay deployment resumes after approximately 118 hours of computation, while the co-evolutionary one resumes ten hours later. Moreover, in both cases the execution is faster than the execution of B&B executed alone. This demonstrate that metaheuristics allow by far to speed up the execution of exact methods.

6. Conclusions and future work

We have proposed a parallel hybrid combinatorial optimization approach which combines two metaheuristics - a genetic algorithm and a memetic algorithm, and an exact method - a B&B algorithm. In addition to their efficiency in finding the optimal solutions, both metaheuristics bring to the new method their capabilities of exploration and intensification of the search process. On the other hand, the B&B algorithm contributes with its ability to provide optimal solutions. Both metaheuristics are combined in a high level co-evolutionary mode in order to obtain a new hybrid metaheuristic, called AGMA [1]. This latter is combined with the B&B algorithm either in a relay

No. of islands	1	10	20	30	40	50	60-100
S-metric	1086366	1123495	1123602	1123519	1123617	1123617	1123654(Exact)

Table 4. S-metric value according to the number of islands

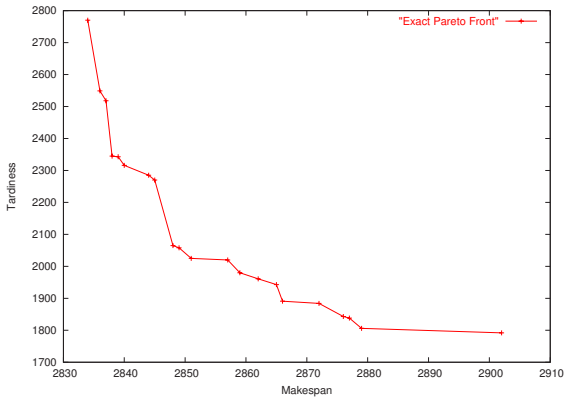

Figure 1. The exact obtained Pareto front

mode or in a co-evolutionary mode in order to build a new exact method.

The parallelization of this method on a grid is performed by exploiting three well-known parallel models - the island model for the GA, the multi-start model for the local search part of the MA and the parallel tree exploration model for the B&B algorithm. The implementation of the three parallel models is based on the coordination model proposed in [7].

The method has been experimented on a computational grid composed of more than 400 machines belonging to four distinct domains. The experiments lasted several days allowing to solve a bi-objective permutation flow-shop instance which has never been solved. The experimental results demonstrate the effectiveness of the approach and its efficient mechanisms: load balancing, fault-tolerance, granularity management and termination detection.

The analysis of these results raises new interrogations on hybridization and parallel computing. Indeed, regarding hybridization we plan to evaluate the separate contribution of each individual method to the effectiveness. Moreover, it is important to study in the co-evolutionary mode the distribution of resources between the two methods: exact methods and metaheuristics. On parallel computing, questions concern the behavior and limits of the method on a larger computational grid and more complex instances. To provide answers to these questions, we plan to use the Grid5000 (https://www.grid5000.fr) experimental grid in the near future.

References

[1] M. Basseur, F. Seynhaeve, and E.-G. Talbi. Adaptive mechanisms for multi-objective evolutionary algorithms. In *Congress on Engineering in System Application CESA'03*, pages 72–86, Lille, France, 2003.

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

[3] D.Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems*, 7:80–112, 1985.

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

[5] G. Fedak, C. Germain, V. Neri, and F. Cappello. XtremWeb: building an experimental platform for Global Computing. *Workshop on Global Computing on Personal Devices (CCGRID2001), IEEE Press*, May 2001.

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

[7] M.-S. Mezmaz, N. Melab, and E.-G. Talbi. Towards a Coordination Model for Parallel Cooperative P2P Multi-objective Optimization. In *In Springer Verlag LNCS 3470, Proc. of European Grid Conf. (EGC'2005)*, pages 305–314, Amsterdam, The Netherlands, 14–16 Feb. 2005.

[8] E.-G. Talbi. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics, Kluwer Academic Publishers*, Vol.8:541–564, 2002.

[9] E.-G. Talbi, E. Alba, N. Melab, and G. Luque. *Metaheuristics and Parallelism*, chapter 4, pages 79–103. In Wiley Book On Parallel Metaheuristics: A New Class of Algorithms, 2005.

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