A Cooperative Tree-based Hybrid GA-B&B Approach for Solving Challenging Permutation-based Problems

Malika Mehdi
6, rue Richard
Coudenhove-Kalergi
L-1359, Luxembourg
Luxembourg
malika.mehdi@uni.lu

Jean-Claude Charr
INRIA Lille Nord-Europe, 40
avenue Halley
Bt. A, Park Plaza 59650
Villeneuve d'Asq, France
jean-claude.charr@univ-fcomte.fr

Nouredine Melab
INRIA Lille Nord-Europe, 40
avenue Halley
Bt. A, Park Plaza 59650
Villeneuve d'Asq, France
Nouredine.Melab@lifl.fr

El-Ghazali Talbi
INRIA Lille Nord-Europe, 40
avenue Halley
Bt. A, Park Plaza
59650 Villeneuve d'Asq,
France
El-Ghazali.Talbi@lifl.fr

Pascal Bouvry
6, rue Richard
Coudenhove-Kalergi
L-1359, Luxembourg
Luxembourg
pascal.bouvry@uni.lu

ABSTRACT

The issue addressed in this paper is how to build low-level hybrid cooperative optimization methods that combine a Genetic Algorithm (GA) with a Branch-and-Bound algorithm (B&B). The key challenge is to provide a common solution and search space coding and associated transformation operators enabling an efficient cooperation between the two algorithms. The tree-based coding is traditionally used in exact optimization methods such as B&B. In this paper, we explore the idea of using such coding in Genetic Algorithms. Following this idea, we propose a pioneering approach hybridizing a GA with a B&B algorithm. The information (solutions and search sub-spaces) exchange between the two algorithms is performed at low-level and during the exploration process. From the implementation point of view, the common coding has facilitated the low-level coupling of two software frameworks: ParadisEO and BOB++ used to implement respectively the GA and the B&B algorithms. The proposed approach has been experimented on the 3D Quadratic Assignment Problem. In order to support the CPU cost of the hybridization mechanism, hierarchical parallel computing is used together with grid computing.

Categories and Subject Descriptors

D.2.8 [**Combinatorics**]: Permutations and combinations, Combinatorial algorithms; G.1.6 [**Optimization**]: Global optimization

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'11, July 12–16, 2011, Dublin, Ireland.
Copyright 2011 ACM 978-1-4503-0557-0/11/07 ...$10.00.

General Terms

Algorithms

Keywords

Hybrid methods; Branch-and-Bound; Genetic Algorithms; Q3AP; Permutation Problems

1. INTRODUCTION

Genetic algorithms (GAs) [8] have been combined with different optimization methods (heuristic or exact ones) in order to obtain a new class of methods, that explore more efficiently the search space of large and complex combinatorial optimization problems. The basic idea of such hybrids is to take advantage from the complementary of the methods they combine. In this paper, the focus is set on combining GAs and the branch-and-bound algorithm (B&B) [13]. This later uses a tree structure to progressively cover the search space and uses relaxation techniques to eliminate sub-trees that are not likely to lead to optimal solutions.

Relevant ways to combine these two algorithms have been largely discussed in the literature and successfully applied to different classes of optimization problems ([17], [7], [3], [5]). Those methods could be classified in two categories: relay methods working in a pipelined mode (the input of one method is the output of the previous method) or cooperative methods where the two algorithms are executed in a parallel or intertwined way and exchange information to guide their respective searches. Generally, the main information exchanged in a cooperative hybridization between GAs and the B&B algorithm is new best found solutions and the initialization of the GA's population from the B&B's pool of nodes. However, the advantages took from the cooperation between GA and B&B could go beyond the traditional weakness/strength balance, best solutions exchanges, etc., if the characteristics of the two algorithms are exploited to find some "connections" between their respective searches.

The idea in this paper is to reuse the B&B tree-based representation in GAs in order to design a tight cooperating strategy. Indeed, special tree-based crossover and mutation operators are introduced in order to control the GA's search in unexplored areas, not yet visited by the B&B and exclude pruned sub-spaces from the search. In addition, the best individuals in GA's final populations are used by the B&B to localize near-optimal solutions. This new cooperative scheme is based on a coding approach of the B&B tree proposed in [15] and applied to permutation problems.

Applications of permutation problems could be found in different areas: assignment problems (which alone include a variety of design problems in different areas), scheduling and production sequencing problems, classical traveling salesman problem etc.

In this paper, the new hybrid cooperative scheme GA-B&B is evaluated on standard benchmarks of the 3 dimensional quadratic assignment problem (Q3AP) [10], one of the hardest permutation problems. The Q3AP arises in a wireless communication design problem used in the HARQ protocol (hybrid automatic repeat request) and whose solution can significantly increase throughput and reduce the cost for providing reliable digital transmission over noisy fading channels. The largest Q3AP benchmark solved to optimality is of size 14 [6] but real life application sizes in the telecommunication field are of size 16, 32 and even 256!. Solving large benchmarks of this problem to optimality is a big challenge for exact algorithms. Thus, metaheuristics are more suitable for large instances of the problem. However, hybrid methods combining both exact search algorithms and metaheuristics may help to solve larger instances to optimality or to provide near-optimal solutions of better quality.

When applied to large benchmarks of this problem, the hybrid method GA-B&B proposed in this paper is CPU time intensive. Therefore, in order to speed up the execution, we proposed a hierarchical master/slave parallelization on hundreds of processing cores belonging to a computational grid (Grid'5000), to solve benchmarks up to size 15.

The rest of the paper is organized as follows: in Section 2, GA and the B&B algorithm are presented, and some insights are given on hybrid methods combining them. The new hybrid method is described in Section 3 and in Section 4 we present the frameworks used to implement the cooperative optimization method and the tackled problem, Q3AP. In Section 5, the results of the experiments conducted using Grid'5000, are reported. Finally, concluding remarks and future research perspectives are drawn in Section 6.

2. RELATED WORKS

2.1 Genetic algorithms

Genetic Algorithms (GAs) [8] are a very popular class of evolutionary algorithms (EAs). They are stochastic search and optimization techniques inspired from the theory of evolution and the adaptation of species. Techniques inspired by natural evolution such as inheritance, mutation, selection, and crossover, are used to generate fitter individuals through generations (see Algorithm 1). As population-based metaheuristics, GAs are exploration-oriented search algorithms: the use of multiple starting search points (the individuals of the population) allows a diversification of the search. On the other hand, it is acknowledged that integrating a local search during the different steps of a GA, enables to exploit promising solutions, as local search methods are known for their intensification capacities. This kind of hybridization, commonly known as memetic algorithms or hybrid genetic algorithms in the literature [7] [17], allows to balance between diversification and intensification phases. GAs have also been associated with exact methods. The advantages taken from a hybridization with an exact algorithm goes beyond the traditional balance between diversification and intensification. Indeed, if we consider the B&B algorithm for discrete combinatorial optimization problems, the search space is explored in a different way compared to GAs or to metaheuristics in general. A tree is used to progressively enumerate all the solutions of the search space. This representation allows a complete control of the search space, as large as it can be:

- A memory module allows one to memorize the explored sub-spaces (nodes in the tree), those sub-spaces will not be re-explored,
- The search can be concentrated and maintained in a given sub-space (sub-tree).

All those characteristics, if re-used in GAs, will allow a tight cooperation with tree-based exact methods. For example, special GA operators could use the tree-representation and the B&B's memory in order to maintain the search exclusively in the non explored sub-spaces. The basics of the B&B algorithm are given in the next section.

Algorithm 1 Template of an evolutionary algorithm

```
Generate initial population: P(0)
t ⇐ 0
while (not stop_creterion(P(t)) do
  Evaluate (P(t))
  P'(t) ⇐ Selection(P(t))
  P'(t) ⇐ ReproductionOps(P'(t))
  P(t+1) ⇐ Replacement(P'(t), P(t))
  t ⇐ t+1
end while
```

2.2 The Branch-and-bound algorithm

The branch-and-bound (B&B) algorithm [13] 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 sub-problems of the total solution space. The construction of such a tree and its exploration are performed as follows: two containers are allocated. The first one stores the best solution found so far (called the upper-bound) and it is initialized to ∞, and the second one stores the set of yet unexplored sub-problems and initially stores the root node. At each iteration, a sub-problem is selected according to some criterion (bound value of the sub-problem, depth of the sub-problem) from the container of yet unexplored sub-problems. Then, a "branching" operation is applied on the selected sub-problem, subdividing the sub-problem's solution space into two or more subspaces to be investigated in a subsequent iteration. For each one of these, it is checked whether the subspace consists of a single solution, in which case it is compared to the best solution found so far and the best one is

kept (updating the upper-bound). Otherwise, a "bounding" function for the subspace is calculated and compared to the upper-bound. If the bound of the subspace is higher than the upper-bound (in the case of a minimization problem) which means that all the solutions contained in this subspace have higher costs than the best solution found so far, the whole subspace is discarded. Otherwise, the subspace and its bound are stored in the container of yet unexplored sub-problems. This process is repeated until the container of yet unexplored sub-problems is empty.

2.3 Hybrid methods

Approaches combining exact methods and metaheuristics have been largely studied in the literature ([17], [7], [3], [5]). A review of the most important works in this field could be found in [16] and a classification could also be found in [11]. In this paper, the focus is set on hybrid methods combining GAs and the B&B algorithm. If we consider the way the methods interact with each other (design point of view), we may distinguish between:

- Sequential execution (or relay): each method uses the output of the previous one as its input, acting in a pipeline fashion.
- Cooperative (teamwork): many cooperating search agents evolve in parallel, each agent carries out a search in a solution space and cooperates with the other agents to find the global optimum.

The most important issue in a cooperative search is the nature of the information exchanged between the two methods. In many works from the literature, the information exchanged from the B&B to the GA is either the initial populations (taken from the B&B's pool of promising nodes to explore (as in [7], [5] and [3]), and/or the best found solutions to integrate in the GA's population as in [17]. In [17], a Branch-and-cut (B&C) algorithm is cooperating with a memetic algorithm. The B&C also sends the current dual variable values to the GA each time a new incumbent solution is found, in order to lead the GA to a better global solution. Alternatively, the information exchanged from the GA to the exact algorithm is first of all, new upper-bounds. This is generally the only information exchanged from the GA to the B&B in most of the reviewed works from the literature. In [5], if the best solution returned by the GA improves the upper-bound of the B&B, this solution will be grafted onto the B&B tree and if it contains variables that are not yet branched, new nodes are built in the tree and the B&B solves them immediately. A similar idea is used in the cooperation strategy proposed in this paper, for permutation-based problems.

Indeed, the new cooperative scheme uses a special representation of the search space, the coded B&B tree [15], in both sides GA and B&B. Using the same representation, the exchanged information between the two algorithms is no longer limited to initial populations (from the B&B to the GA) and best found solutions (from the GA to the B&B). The principles of the new scheme are explained in the next section.

3. A NEW HYBRID GA-B&B COOPERATIVE SCHEME FOR PERMUTATION-BASED PROBLEMS

In this section, we present the new cooperative hybrid GA-B&B scheme. To make this paper self-sufficient, we need to introduce the coded B&B tree representation of the search space [15]. Then, the new hybridization scheme is presented as well as the special transformation operators used in the GA.

3.1 The coded tree-based representation of the search space

In order to enumerate all the solutions of the search space, the B&B algorithm progressively builds a tree that covers the search space. The root node is the initial problem to be solved of size N and intermediate nodes are sub-problems of size $N - d$ (d is their depth in the tree). Finally, the leaves represent complete solutions of the problem. In [15], a special coding for this tree is used to facilitate the decomposition of the search space between several B&B processes. Indeed, each tree node is assigned a unique number (identifier) and any set of contiguous nodes (having successive numbers) could be represented as an interval.

The node's path, node's weight and node's rank define the number associated to any node in the tree. The number associated to any node in a m-permutation tree (a tree used to represent the search space of a permutation-based problem where a solution is represented by m permutations) is calculated using Equation (1) where $path(n)$ is the set of nodes from the root to the node n, including both the root and n, $weight(n)$ is the number of leaves of the sub-tree of the node n computed by Equation (2), and $rank(n)$ is the position of a node n among its sibling nodes. Using this approach, any set of contiguous nodes can be represented by an interval as shown in Figure 1, which is an example for coding a one-permutation tree of size $N = 3$. Assuming that m is the number of permutations in the tackled problem and N is the size of the permutations, the size of the search space is $S = N!^m$ and the global search space can be represented by the global interval $I = [0, N!^m[$.

$$number(n) = \sum_{i \in path(n)} rank(i) * weight(i) \quad (1)$$

$$weight(n) = (N - depth(n))!^m \quad (2)$$

$$range(n) = [number(n), number(n) + weight(n)[\quad (3)$$

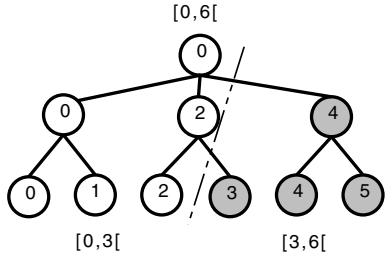

Figure 1: The tree-based representation where each node has a unique number and contiguous nodes are represented by intervals

In Figure 1, the global interval is equal to $[0, 6[$. Finally, the notion of range of n, noted as $range(n)$, defines the interval that contains all the nodes in which the node n is the root node (see Equation (3)). This method is used in [15] to share the global search space interval between several B&B processes. Each process will get a sub-interval to solve. A special operator named "unfold" is used to transform any interval to a minimal set of nodes in each B&B process.

3.2 Basics of the new hybrid GA-B&B cooperative scheme

The new hybrid GA-B&B cooperative scheme is designed as an island couple: a GA island and a B&B island. The two islands are able to asynchronously exchange information. The B&B island holds three pools: a pool of yet unexplored intervals (initialized with the global interval at the starting of the application), a pool of nodes to solve (generated from the first pool) and a pool of promising nodes (empty at the beginning of the method). The GA island contains a population of individuals that is initialized from the global unexplored interval. After the initialization step, both islands begin exploring in parallel their search spaces using the B&B algorithm or the GA. The B&B island starts solving sequentially the nodes in its pool. When a node is solved to optimality, its *range* (the sub-interval containing all the solutions for whom the solved node is root) is deleted from the list of yet unexplored intervals.

On the other hand, the GA evolves its population until a given number of generations. This parameter represents the migration frequency between the GA island and the B&B island. Afterward, the GA sends its final population to the B&B island in order to inform the B&B algorithm of the best solution it has found and to point out promising regions to the B&B algorithm. Indeed, when the B&B island receives the population, it compares the best solution in this population to its upper-bound and if it is fitter than the upper-bound, the upper-bound is replaced by the new best solution. Moreover, for each different solution taken from the GA's population, a deviation, equal to the difference between the fitness value of the solution and of the best solution value found so far, is computed. If the deviation Δ is less than a threshold $\Omega \simeq 30\%$ of the upper-bound, this solution is used to identify promising regions by choosing its largest ancestor belonging to one of the yet unexplored intervals. This ancestor is stored in the pool of promising nodes because its range may contain good solutions for the problem. Thus, the B&B interrupts its sequential exploration of the search space and starts solving the promising nodes first. Therefore, the GA is not limited to finding a global optimum to the problem.

Finally, the B&B generates a new initial population from an unexplored interval and sends it to the GA. If the pool of intervals is empty, if the biggest interval is too small or if the time limit predefined by the user is reached, the GA gets an empty population as a stopping criterion. The new population, sent to the GA, represents a new sub-space to explore by the GA and the GA's search is limited to that sub-space with the use of special transformation operators (see Section 3.3). The cooperation steps between the two islands are illustrated in Figure 2.

3.3 The GA's side: interval-based transformation operators

The transformation operators (crossover and mutation) used in the GA are made in such a way that the resulting offspring through generations do not fall outside the subspace delimited by the interval used to initialize the GA's first population.
The crossover proposed in this paper is an adaptation of the one point Partially Mapped Crossover (PMX) [9]. This latter consists in randomly selecting a cutting point in the two parents. Then, the first part of the first offspring C_1 (respectively the second offspring C_2) is inherited from the first part of the first parent P_1 (respectively the second parent P_2) and the remaining positions are completed from the appropriate alternate parents.

In order to control the behavior of this operator, we proceed as illustrated in Figure 3. First of all, the GA is given a sub-interval representing the search sub-space to be explored. When the operator is applied on a pair of individuals (parents), each parent is mapped to the tree-based representation by generating its associated number in the tree. Then, the ranges associated to all intermediate tree nodes that lead to the considered individual (which is also a final solution in the tree) are computed. For example, in Figure 3 for the node P, the ranges of all its predecessors, represented by the blue intervals, are computed. Next, the node with the largest range and whose range is included in the initial interval given to the GA is chosen (in Figure 3 it's the predecessor of P with $depth = 1$). The depth of this node will be used as a minimum cutting point in the PMX crossover. Indeed, if only genes after this cutting point are modified, the resulting offspring will not fall outside the range of the selected node and consequently it will be inside the GA's interval. This procedure is not much time consuming because the determination of the appropriate cutting limit is done only once, when the initial population is given to the GA. This point is inherited by the next generations.

The mutation operator has also been modified using the same strategy. For example, if we consider a pairwise exchange (swap) mutation operator, two points are selected at random and the genes in these positions are swapped. The depth of the largest node included in the interval is used as a lower bound for the swapping points in the individual to mutate.

3.4 Parallelization of the hybrid scheme GA-B&B

The cooperative GA-B&B algorithm is CPU time intensive when used to solve large benchmarks of permutation-based problems. Therefore, the benefits of parallelizing this method are obvious. A hierarchical master/slave parallelism is used. The later is composed of two levels of parallelism. The first one (Figure 4) consists on using the master/slave model in each island (B&B and GA).

The GA uses additional resources (slaves) in order to parallelize the evaluation and the transformation tasks while the B&B uses additional resources in order to solve a group of nodes simultaneously. Each B&B slave includes a B&B solver that simply receives one node to be solved and returns the result to the B&B island.

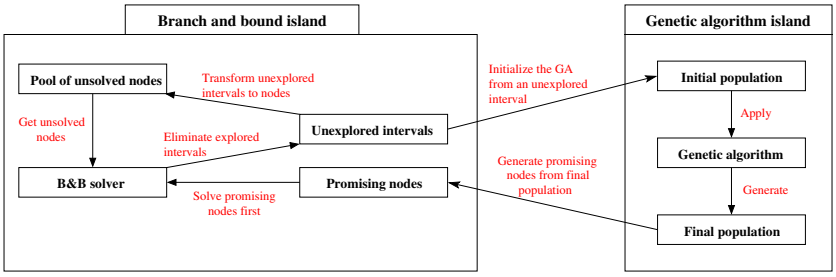

Figure 2: Data exchanged between the pair of B&B and GA islands in the cooperative model

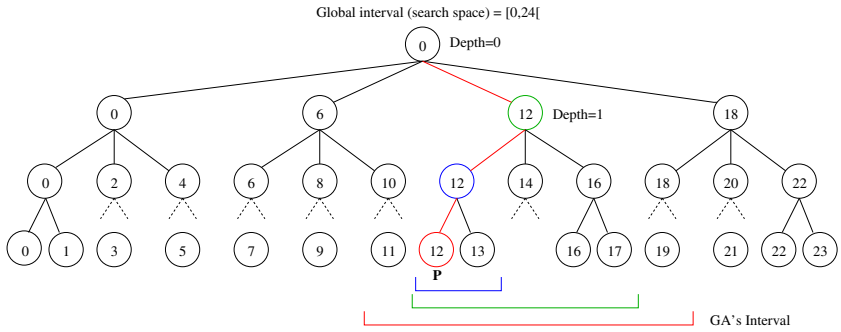

Figure 3: Selecting the cutting point limit in the interval-based crossover and mutation operators

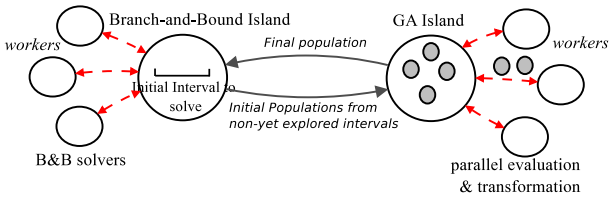

Figure 4: The first level of parallelism in the hybrid GA-B&B: a master-slave model on the top of each island

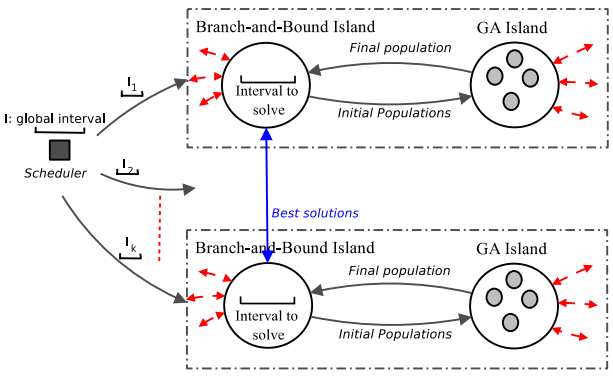

Figure 5: The second level of parallelism in the hybrid GA-B&B: a hierarchical master-slave model

The second level of the parallelism is illustrated in Figure 5 and consists of dividing the global search space, represented by the global interval I, into k equal sub-intervals $\{I_1, I_2, \ldots, I_k\}$. These sub-intervals will be distributed among k couples of islands GA-B&B. The B&B islands in all the couples GA-B&B are connected to each other via a complete topology in order to exchange their best found solutions. On the other hand, the GA islands do not exchange data with each other.

4. IMPLEMENTATION

In this section, we present the two frameworks used in the implementation of the hybrid method: ParadisEO and BOB++.

4.1 ParadisEO framework

The GA has been implemented using ParadisEO (PARAllel and DIStributed Evolving Objects) [2] which is a C++ open source framework dedicated to the reusable design of parallel hybrid metaheuristics. ParadisEO provides a broad range of features for solving optimization problems, such as evolutionary algorithms, local search methods, different hybridization mechanisms for metaheuristics, etc. It also includes the most common parallel and distributed models adapted to distributed-memory machines and shared-memory multiprocessors, as they are implemented using standard libraries such as MPI and PThreads.

4.2 BOB++ framework

The B&B algorithm used in our experiments for the Q3AP is developed using the BOB++ framework, this algorithm could be found in [6]. BOB++ [4] is a C++ framework dedicated to the design of exact optimization methods for solving combinatorial optimization problems on parallel architectures. The parallelization is done using POSIX threads. However, the framework is easily expendable and other parallelization models are under development. Moreover, BOB++ provides many search algorithms (such as B&B, divide and conquer,...) applied to many problems (e.g. TSP, QAP, and Q3AP).

5. EXPERIMENTS

In order to evaluate the hybrid model presented in this paper, we have applied it to the three dimensional quadratic assignment problem (Q3AP [12]) and solved many instances of this problem over Grid'5000. In the following section, we present this problem and the different instances tackled in our experiments.

5.1 The three dimensional quadratic assignment problem

This problem was initially introduced to model a problem arising in data transmission system design. More exactly the problem of finding optimal mappings for two successive transmissions in the hybrid automatic repeat request protocol (HARQ). For each transmission, a mapping should be used between N QAM (quadratic amplitude modulation) symbols and N segments of the message to send. To each assignment is associated a probability of error related to the noise of the channel. Those probabilities are stored in a matrix C. Diversifying the mappings in each (re)transmission helps to globally minimize the bit-to-error rate (BER). Solving the Q3AP consists in finding an optimal or a good solution minimizing the BER in the two mappings. A solution for this problem is represented by two permutations of size N. Thus, the solution space size is $N!^2$. The objective function is given by Equation (4). Equations (6), (7) and (8) express the uniqueness constraints (one symbol to one segment and vice versa in the two mappings).

$$min \left\{ \begin{array}{l} \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{N} b_{ijp}x_{ijp} \; + \\ \\ \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{N}\sum_{k=1}^{N}\sum_{n=1}^{N}\sum_{q=1}^{N} C_{ijpknq} \; x_{ijp} \; x_{knq} \end{array} \right. \quad (4)$$

$$x \in I \cap J \cap P,\; x = 0, 1 \quad (5)$$

With: I, J, P sets of same cardinality N, and:

$$I = \left\{ \; x \geq 0 : \sum_{j=1}^{N}\sum_{p=1}^{N} x_{ijp} = 1 \quad for\; i = 1, .., N \right. \quad (6)$$

$$J = \left\{ \; x \geq 0 : \sum_{i=1}^{N}\sum_{p=1}^{N} x_{ijp} = 1, \quad for\; j = 1, .., N \right. \quad (7)$$

$$P = \left\{ \; x \geq 0 : \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ijp} = 1, \quad for\; p = 1, .., N \right. \quad (8)$$

Note that the artificial Q3AP benchmarks could be obtained from QAP ones using Equation (9). F and D are the two matrices representing the flues and the distances between factories/sites in the QAP. C is the data matrix of the Q3AP.

$$C_{ijpknq} = F_{ik} * D_{jn} * F_{ik} * D_{pq}; (i, j, p, k, n, q = 1..N) \quad (9)$$

The only exact method used for this problem in the literature is the B&B algorithm proposed in [12] and a parallel version for shared memory architectures in [6]. The first heuristic methods used to solve Q3AP are reported in [10]. These algorithms are derived from the most successful methods used to solve the QAP (simulated annealing algorithm, fast ant colony algorithm (FANT), etc.). Parallel genetic algorithms have also been used in some works to solve larger benchmarks [14].

5.2 Grid5000 experimental testbed

We conducted the experiments over Grid'5000 which is the French nation-wide experimental grid that inter-connects 9 sites via RENATER (the French academic network). Currently, the GRID is composed of more than 6200 cores. The inter-connections sustain communications of 10 Gbps.

5.3 Global settings

To evaluate the performances of the new hybrid scheme, our experimental process is composed of several steps detailed in the following sections. The Q3AP benchmarks used are all derived from standard QAP benchmarks which could be found in the QAPLIB [1]. Some benchmarks are generated from other instances of larger size in the QAPLIB: Nug12c is generated from Nug13 in the QAPLIB (one column and one row form each matrix of the Nug13 are withdrawn), Had12b is obtained after permuting the two matrices of the Had12, Had15 is generated from Had16 and Nug15a from Nug16a. The parameters used in the GA islands of the cooperative model are listed in Table 1. The migration frequency between a couple of islands (a B&B island and a GA island) was fixed to 200 generations: after 200 generations, each GA sends its final population to the B&B island and gets a new initial population. These parameters were chosen because after many sets of experiments conducted on many benchmarks they proved to be sufficient for the convergence of the GA.

Table 1: Global settings used for the islands of GA

Pop.Size	Cross.Rate	Mut.Rate
60	1	0.6
Mig.Freq	**Nb.Migrants**	**Stop.criterion**
200	60	empty pop.received

5.4 Evaluation of the performances of the GA-B&B algorithm against the B&B alone

In this stage, the two algorithms GA-B&B and B&B alone are executed for 2 hours on the following Q3AP benchmarks: Had12, Had12b, Nug12, Nug12c. The objective of this step of the experiments is to explain the behavior of the two algorithms and show the benefits gained from the hybridization with the GA.

The machine used in this set of experiments is a 4 Dual Core CPU, 2.5 GHZ, 64 bits. The results are reported in Table 2. In order to give the two models (GA-B&B and B&B alone) equal chances, the same initial solutions are used in both of them (column 3). Since the B&B is deterministic we need only one run for this model. On the other hand, in the cooperative model (GA-B&B), the global behavior of the hybrid algorithm is stochastic. Thus, we need several runs for this model. The reported results are the average values among 8 runs. The experimental results show that the hybrid GA-B&B model finds a better solution than the B&B for the benchmarks Had12 and Nug12c. Equal solutions are found for Had12b and Nug12 but in less time while using the GA-B&B.

In addition, we expected that the hybrid cooperative model explores more space than the B&B alone because of the earlier improvement of the upper-bound in the B&B solver (in the B&B algorithm the best found solution is the upper-bound and the better is the upper-bound the less it takes time to solve the nodes because of the pruning operations). The last column of Table 2 represents the percentage of the big interval (representing the complete search space) that is not yet explored. We notice an important improvement made by the GA-B&B according to the number of the explored nodes and to the percentage of unexplored space, in the benchmarks Nug12 and Nug12c: the unexplored interval is 61% of the global interval with the hybrid model and 83% with the B&B alone for Nug12, and 76% vs 95% for Nug12c. This impact is less important with the benchmarks Had12 (88% vs 90%) and not apparent with the benchmark Had12b where the two models have explored the same percentage of the search space. From this step of the experiments we clearly notice that the use of the GA island allows the B&B to find the optimal solutions more quickly. However, the impact on the execution speed is not always significant. The next step of the experiments consists in evaluating the two models: B&B and the hybrid GA-B&B while solving larger benchmarks but in parallel on the computational grid.

5.5 Evaluation of the performances of the parallel hybrid GA-B&B against the parallel B&B on the grid

In this step of the experiments, both methods (B&B alone and GA-B&B) were given equal execution time periods (2 hours or 5 hours for large instances) to solve each benchmark using the same parallel hierarchical master/slave model on the same number of machines (from Grid'5000: 4 Dual core CPUS, 2.5 GHZ, 64 bits). In Table 3, the number of islands (column 3) represents the number of B&B islands used in the parallel B&B alone and the number of B&B plus GA islands (each B&B island is collaborating with one GA) used in the parallel hybrid GA-B&B model. We notice that after the maximum execution time (2 or 5 hours), the B&B alone was not able to provide a better solution while in the hybrid model (GA-B&B) a better solution was found for all benchmarks. In order to check the impact of the initial upper-bound on the percentage of unexplored space, we used ∞ as a starting solution for the last benchmark: Nug15. Nevertheless, the percentage of the unexplored search-space is the same for both models. At this scale, it is difficult to measure the impact of each factor on the speed up.

5.6 Proving the optimality of the solutions found by the GA-B&B

After comparing the performances of the two algorithms (GA-B&B and B&B alone) we are interested in solving to optimality some benchmarks and hopefully, prove the optimality of some of the solutions found previously by the GA-B&B. For this step, the hybrid GA-B&B could also be used as an exact method and the stopping criterion is the end of the exploration in all the couples GA-B&B. However, since the objective in this step is only to prove the optimality of the best found solutions by the B&B-GA, the order used to solve the remaining sub-spaces by the B&B solvers is not important. Therefore, the cores occupied by the GA islands should be exploited by the B&B islands. In addition, the sub-spaces solved by the GA-B&B model in the previous stage are not re-explored in this step. The results of these experiments are given in Table 4, the best found solutions for the benchmarks Had12, Nug12c, Had12b, Had14 and Nug15 are optimal.

Table 4: Solving some benchmarks to optimality

Bench.	Initial sol.	Optimal	Time(s)	Cores
Had12	19430	19430	2008	440
Had12b	21128	21128	67	440
Nug12c	1326	1326	7429	200
Had14	37598	37598	18h42mn	528
Nug15	2230	2230	9 days 7h	300

6. CONCLUSION AND PERSPECTIVES

In this paper, a new cooperative hybrid scheme combining genetic algorithms and the branch-and-bound (B&B) is proposed to solve large benchmarks of permutation-based problems. The particularity of the new scheme is the use of the same representation of the search space in both algorithms, a coded B&B tree, in order to enable a tight cooperation between them. Because such method is CPU time intensive when applied to large permutation problems, a hierarchical master/slave parallelism is used to speed up the execution. A set of experiments on Q3AP benchmarks are conducted over a computational grid and the performances of the hybrid scheme GA-B&B in localizing optimal solutions are compared to the performances of the B&B when used alone. The objectives of these experiments are to assess the efficiency and effectiveness of the cooperation between the two algorithms. The obtained results indicate that the proposed cooperation strategy enables the B&B to localize optimal or near-optimal solutions more quickly in huge search spaces. In addition, some of the obtained best solutions by the hybrid GA-B&B are proved to be optimal. Indeed, unsolved benchmarks of the Q3AP up to size 15 (Had12, Nug12c, Had12b, Had14 and Nug15), are solved to optimality using the parallel version of the hybrid algorithm, over a computational grid (Grid'5000).

One of the challenging perspectives of this work is to determine the minimum percentage of search space that need to be explored by the hybrid cooperative scheme before one could stop the GA islands and continue only with the B&B to prove the optimality of the best found solution.

Table 2: Results obtained on small Q3AP benchmarks (*the best found value for Nug12 is optimal [12])

Bench.	Algo.	Initial sol	Cons. nodes	Best found	Average time(s)	Left job (%)
Had12	GA-B&B	19600	17	**19430**	4099	87
	B&B	19600	14	19600	7216	90
Had12b	GA-B&B	21190	122	**21128**	**1374**	15
	B&B	21190	122	**21128**	1763	15
Nug12	GA-B&B	680	28	**580***	**1999**	61
	B&B	680	23	**580***	3442	83
Nug12c	GA-B&B	1328	6	**1326**	2331	76
	B&B	1328	5	1328	8171	95

Table 3: Results obtained using the parallel master-slave model on both the hybrid GA-B&B and the B&B algorithms (*the best found value for Nug15 corresponds to the best known value from the literature [12])

Bench.	Algo.	Islands	Cores	Initial sol.	Best found	Time(s)	Left job %
Nug14	GA-B&B	14	200	8140	**8084**	4652	98
	B&B	7	200	8140	8140	8040	98
Had14	GA-B&B	14	256	37726	**37598**	4222	96
	B&B	7	256	37726	37726	7315	96
Nug15a	GA-B&B	18	464	12952	**12776**	7011	99
	B&B	9	464	12952	12952	7614	99
Had15	GA-B&B	18	640	43396	**42860**	2690	99
	B&B	9	640	43396	43396	19092	99
Nug16a	GA-B&B	16	640	15372	**15132**	11155	88
	B&B	8	640	15372	15372	18013	88
Nug15	GA-B&B	30	720	∞	**2230***	533	99
	B&B	15	720	∞	2620	18000	99

Acknowledgments

The present project is supported by the National Research Fund, Luxembourg. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed. The authors would like to thank Bertrand Le Cun, François Galea and Van-Dat Cung for providing us with the BOB++ framework and the B&B algorithm for the Q3AP, also for their technical support.

7. REFERENCES

[1] R. E. Burkard, S. E. Karisch, and F. Rendl. QAPLIB - a quadratic assignment problem library. *European Journal of Operational Research*, 55:115–119, 1991. QAPLIB is found on the web at http://www.seas.upenn.edu/qaplib.

[2] S. Cahon, N. Melab, and E.-G. Talbi. ParadisEO: a Framework for the Reusable Design of Parallel and Distributed Metaheuristics. *Journal of Heuristics*, 10:353–376, 2004.

[3] J.-L. B. Cedric Pessan and E. Neron. Genetic branch-and-bound or exact genetic algorithm? In *Lecture Notes In Computer Science*, volume 4926, pages 136–147, 2008.

[4] A. Djerrah, B. L. Cun, V.-D. Cung, and C. Roucairol. Bob++: Framework for solving optimization problems with branch-and-bound methods. In *Proc. of the 15th IEEE Int. Symp. on High Performance Distributed Computing*, 2006.

[5] A. P. French, A. C. Robinson, and J. M. Wilson. Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems. *Journal of Heuristics*, 7(6):551–564, 2001.

[6] F. Galea, P. Hahn, and B. LeCun. A parallel implementation of the quadratic three-dimensional assignment problem using the Bob++ framework. *The 21st Conference of the European Chapter on Combinatorial Optimization*, 2008.

[7] J. E. Gallardo, C. Cotta, and A. J. Fernàndez. On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37:77–83, 2007.

[8] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[9] D. Goldberg and R. Lingle. Alleles, loci, and the traveling salesman problem. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 154–159, Cambridge, 1985.

[10] P. M. Hahn, B.-J. Kim, T. Stutzle, S. Kanthak, W. L. Hightower, Z. D. H. Samra, and M. Guignard. The quadratic three-dimensional assignment problem: Exact and approximate solution methods. *European Journal of Operational Research*, 184:416–428, 2008.

[11] L. Jourdan, M. Basseur, and E.-G. Talbi. Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620 – 629, 2009.

[12] B.-J. Kim. *Investigation of methods for solving new classes of quadratic assignment problems (QAPs)*. PhD thesis, University of Pennsylvania, 2006.

[13] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[14] M. Mehdi, N. Melab, E.-G. Talbi, and P. Bouvry. Interval-based initialization method for permutation-based problems. In *IEEE Proc. of World Congress On Computational Intelligence*, 2010.

[15] 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.*, 2007.

[16] J. Puchinger and G. R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Lecture Notes In Computer Science*, volume 3562, pages 41–53, 2005.

[17] J. Puchinger, G. R. Raidl, and M. Gruber. Cooperating memetic and branch-and-cut algorithms for solving the multidimensional knapsack problem. In *Pro. of the 6th Metaheuristics Int. Conf.*, 2005.