Hybrid Metaheuristics for Multi-objective Optimization

E-G. Talbi
Laboratoire d'Informatique Fondamentale de Lille,
CNRS - INRIA, Cité scientifique,
59655 - Villeneuve d'Ascq cedex - France

Received 30/11/2013; Accepted: 01/08/2014

ABSTRACT

Over the last two decades, interest on hybrid metaheuristics has risen considerably in the field of multi-objective optimization (MOP). The best results found for many real-life or academic multi-objective optimization problems are obtained by hybrid algorithms. Combinations of algorithms such as metaheuristics, mathematical programming and machine learning techniques have provided very powerful search algorithms. Three different types of combinations are considered in this paper to solve multi-objective optimization problems:

- Combining metaheuristics with (complementary) metaheuristics.
- Combining metaheuristics with exact methods from mathematical programming approaches.
- Combining metaheuristics with machine learning and data mining techniques.

1. INTRODUCTION

Over the last two decades, interest in hybrid metaheuristics has risen considerably among researchers in MOP. The best results found for many practical or academic optimization problems are obtained by hybrid search algorithms. Combinations of metaheuristics with mathematical programming and machine learning have provided very powerful search algorithms.

This paper deals with the design of hybrid metaheuristics and their implementation to solve MOPs. A taxonomy of hybrid algorithms is presented in an attempt to provide a common terminology and classification mechanisms.

Corresponding author. El-ghazali.Talbi@lifl.fr; as@essex.ac.uk
This paper was presented at the EGH2013 Workshop, University of Essex, 2013.

The goal of the general taxonomy given here is to provide a mechanism to allow comparison of hybrid algorithms in a qualitative way. In addition, it is hoped the categories and their relationships to each other have been chosen carefully enough to indicate areas in need of future work as well as to help classify future work. Among existing classifications in other domains, one can find examples of flat and hierarchical classifications schemes. The taxonomy proposed here is a combination of these two schemes - hierarchical as long as possible in order to reduce the total number of classes, and flat when the descriptors of the algorithms may be chosen in an arbitrary order. For each type of hybrids, the main ideas in combining algorithms are detailed. Each class of hybrids is illustrated with some examples. A critical analysis is also carried out.

The paper is organized as follows. First, in section 1.1, our concern is hybrid algorithms combining metaheuristics. In section 1.2, the combination of metaheuristics with mathematical programming approaches is considered. Then, in section 1.3 the combination of metaheuristics with machine learning algorithms is addressed.

1.1. Combining Metaheuristics for MOPs

Until the 1990's, the main focus in the metaheuristic field was on the application of pure metaheuristics to MOPs. Nowadays, the use of pure multi-objective metaheuristics is more and more seldom. A skilled combination of concepts of different metaheuristics can provide a more effcient behavior and a higher flexibility when dealing with real-world and large-scale MOPs.

1.1.1. Low-level relay hybrids (LRH)

This class of hybrids represents multi-ob jective hybrid metaheuristics in which a given metaheuristic is embedded into a single solution based metaheuristic (S-metaheuristic). Few examples belong to this class since S-metaheuristics are not well adapted to approximate the whole Pareto set of a MOP into a single run.

An adaptive hybrid metaheuristic: a multi-objective tabu search hyper-heuristic may be used to optimize the use of different S-metaheuristics [8]. This hybrid approach, tested on timetabling and space allocation, uses a tabu list of S-metaheuristics which is updated by adding the last used S-metaheuristic and/or the worst one, in terms of performance. Hence, this hybrid approach will adapt dynamically the search according to the performance of various S-metaheuristics. More efficient multi-objective S-metaheuristics will be more frequently used during the search.

1.1.2. Low-level teamwork hybrids (LTH)

Population based metaheuristics (P-metaheuristics, e.g. evolutionary algorithms, scatter search, particle swarm, ant colonies) are powerful in the approximation of the whole Pareto set while S-metaheuristics are efficient in the intensification of the search around the obtained approximations. Indeed, S-metaheuristics need to be guided to solve MOPs.

Therefore, most efficient multi-objective P-metaheuristics have been coupled with S-metaheuristics such as local search, simulated annealing and tabu search, which are powerful optimization methods in terms of exploitation of the Pareto sets approximations. The two classes of metaheuristics have complementary strengths and weaknesses. Hence, LTH hybrids in which S-metaheuristics are embedded into P-metaheuristics have been applied successfully to many MOPs. Indeed, many state-of-the art hybrid schemes are P-metaheuristics integrating S-metaheuristics.

Multi-objective evolutionary local search algorithm: many multi-objective hybrid metaheuristics proposed in the literature deal with hybridization between P-metaheuristics (e.g. evolutionary algorithms) and S-metaheuristics (e.g. local search). For instance, the well-known genetic local search[1] algorithms are popular in the multi-objective optimization community [28][18][17][13]. The basic principle consists of incorporating a local search algorithm during an evolutionary algorithm search. The local search part could be included by replacing the mutation operator, but it can also be added after each complete generation of the evolutionary algorithm [5]. The classical structure of a multi-objective genetic local search (MOGLS) algorithm is shown in figure 1, which depicts the relationships between the evolutionary multi-objective (EMO) component and the local search one.

The local search algorithm can be applied in a given direction (i.e. weighted aggregation of the objectives) [17]. In order to adapt the basic local search algorithm to the multi-objective case, one may take into account the Pareto dominance relation [5]. The algorithm works with a population of non-dominated solutions *PO*. The hybridization process consists in generating the neighborhood of each solution of the Pareto set approximation *PO*. The new generated non dominated neighbors are inserted into the approximation Pareto set *PO*. Solutions belonging to the Pareto set *PO* and dominated by a new introduced solution are deleted. This process is reiterated until no neighbor of any Pareto solution is inserted into the Pareto set *PO*. The Pareto local search algorithm is described below (Algorithm 1.1.2):

[1]Called also *memetic*

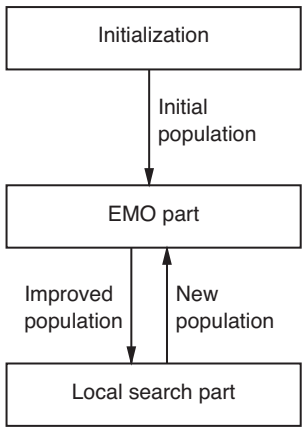

Figure 1. Generic form of multi-objective genetic local search algorithms (MOGLS).

Template of the Pareto guided local search (PLS) algorithm.
Input: an approximated Pareto set *PO*.
Do
$S' = PO$;
Generate the neighborhood PN_x for each solution x of S';
Let PO be the set of non-dominated solutions of $S' \cup PN_x$;
Until $PO = S'$ (the population has reached the local optima);
Output: Pareto set PO.

1.1.3. High-level relay hybrids (HRH)

In HRH hybrids, self-contained multi-objective metaheuristics are executed in a sequence. A classical HRH for MOP is the application of an intensification strategy (e.g. path relinking, S-metaheuristic) on the approximation of the Pareto set obtained by a P-metaheuristic [11][23].

Target aiming Pareto search - the TAPAS algorithm: S-metaheuristics can be combined with any multi-objective metaheuristic to improve the quality of a Pareto approximation. First, a multi-objective metaheuristic (e.g. any P-metaheuristic) is used to generate a good approximation *P* of the Pareto set in terms of diversity. The design of the TAPAS algorithm was motivated by the need to improve the approximation *P* in terms of convergence towards the optimal Pareto set. Indeed, any S-metaheuristic algorithm can be applied to improve the quality of this approximation [22].

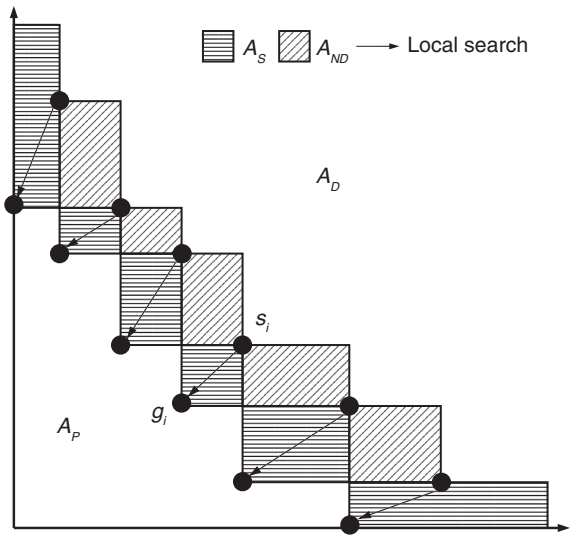

Figure 2. The hybrid TAPAS algorithm for multi-objective optimization: the goal g_i of a solution s_i is defined in function of s_i neighbors in the objective space.

In the TAPAS algorithm, a S-metaheuristic l_i (e.g. tabu search[2]) is applied to each solution s_i of the initial Pareto set P. A specific mono-objective function o_i is defined for each search l_i. The defined objective function o_i must take into account the multiplicity of the S-metaheuristics invoked. Indeed, two S- metaheuristics should not examine the same region of the objective space, and the entire area that dominates the Pareto approximation P should be explored in order to converge towards the optimal Pareto front. The definition of the objective o_i is based on the partition of the objective space O according to the approximation P (see figure 2):

$$A_D = \{s \in O / \exists s' \in P, s' \prec s\}$$

$$A_{ND} = \{s \in O / \forall s' \in P, (s' \not\prec s) \; and \; (s \not\prec s')$$

$$A_S = \{s \in O / \nexists s' \in P, s \prec s'\}$$

$$A_P = \{s \in O / \exists s_1, s_2 \in P, (s \prec s_1) \; and \; (s \prec s_2)\}$$

Each solution $s_i \in P$ is associated with a part A_S^i of A_S. If l_i is able to generate a feasible solution in A_S^i, then the approximation is improved according to the convergence, without decreasing the diversity.

[2]An efficient S-metaheuristic for the target problem should be selected.

To guide the search, a goal g_i is given to each S-metaheuristic l_i, with g_i being the point that dominates all points of A_S^i . In cases where certain coordinates of g_i cannot be defined (e.g. the extremities of P), a lower bound for the missing coordinates should be used. For an objective f_m, the goal g_p is computed as follows:

$$f_m(g_p) = arg \ \ min_{\{f_m(s')/(s' \in P) \ and \ (f_m(s') < f_m(s))\}}(f_m(s') - f_m(s))$$

Then, the objective o_i is stated as follows:

$$min(\sum_{j=1}^{M} | \ f_j(s) - f_j(g_i) \ |^r)^{1/r}$$

When a S-metaheuristic l_i reaches the goal g_i or when it finds a solution that dominates g_i, it stops and produces an archive a_i which contains all the current solutions that are non-dominated. When all the S-metaheuristics l_i are terminated, a new Pareto approximation set P' is formed by the Pareto union of all a_i. Because P' might be improved by another application of S-metaheuristics, the complete process is iterated until P' does not differ from P.

Filling the gap of a Pareto approximation with path-relinking: path relinking can be combined with any multi-objective metaheuristic to intensify the search around a Pareto approximation. First, a multi-objective metaheuristic (e.g. any P-metaheuristic) is used to generate a good approximation of the Pareto set. Then, path relinking concept can be applied to connect the non-dominated solutions of the approximated Pareto set [6][7][19]. The design questions which must be considered are:

- **Selection of the initial and the guiding solutions:** this design question concerns the choice of the pair of solutions to connect. For instance, a random selection from the approximated Pareto set may be applied [6]. Otherwise, some criteria must be used to choose the initial and the guiding solutions: distance between solutions (e.g. distance in the decision or the objective space), quality of the solutions (e.g. best solution according to a reference point or weighted aggregation), etc.
- **Path generation:** many paths may be generated between two solutions. One has to establish which path(s) has to be explored and selected. Among other concepts, a neighborhood operator and a distance measure in the decision space have to be defined. For instance, the shortest paths may be generated according to the selected neighborhood operator [6]. Let us consider x as the current solution and y as the

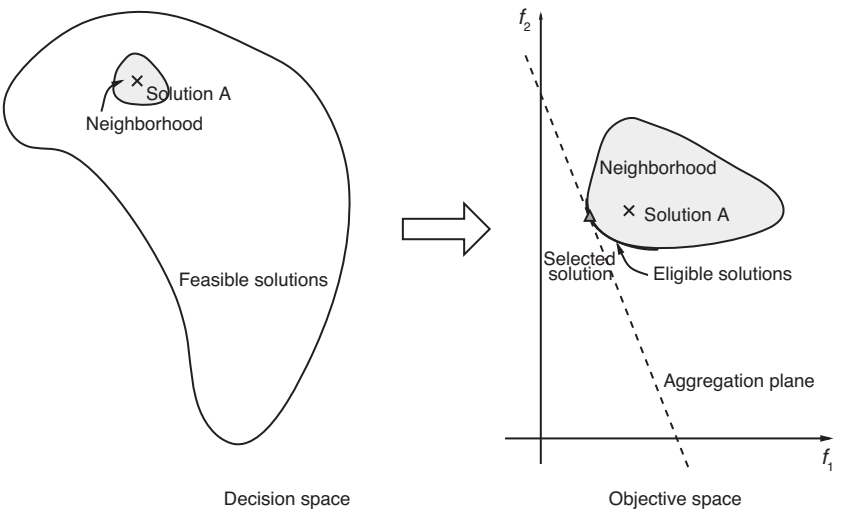

Figure 3. Path Relinking algorithm filling the gap between two non-dominated solutions of an approximation Pareto set: neighborhood exploration.

guiding solution. The neighborhood N of x is generated with the following constraint: $\forall z \in N$, $d(z, x) < d(z, x)$. From this neighborhood, only the non-dominated solutions may be selected to be potential solutions of the future paths (see figure 3). The process is iterated, until a complete path from x to y is generated. Many paths may also be considered. However, generating all possible paths may be computationally expensive. Moreover, the non- dominated solutions may also be selected to participate to a Pareto local search algorithm as shown in figure 4 [6].

1.1.4. High-level teamwork hybrid (HTH)

HTH hybrids scheme involves several self-contained multi-objective metaheuristics performing a search in parallel and cooperating to find a Pareto set approximation.

Cooperative multi-objective evolutionary algorithms: a growing interest is dedicated to design and implement parallel cooperative metaheuristics to solve multi-objective problems. The majority of designed parallel models in the literature are evolutionary algorithms [15] [20] [26] [27]. In multi-objective

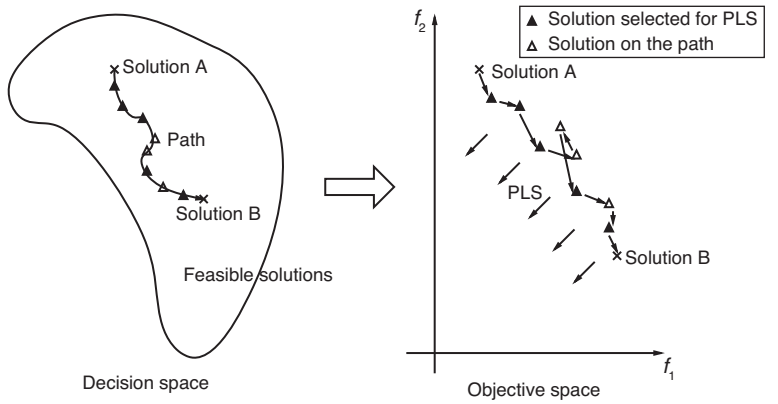

Figure 4. Path relinking algorithm combined with a Pareto local search (PLS) algorithm.

evolutionary algorithms, the individuals are selected from either the population, the Pareto archive or both of them. In the multi-objective island model, different strategies are possible. For instance, the newcomers replace individuals selected randomly from the local population that do not belong to the local Pareto archive. Another strategy consists in ranking and grouping the individuals of the local population into Pareto fronts using the non-dominance relation.

The solutions of the worst Pareto front are thus replaced by the new arrivals. One can also make use of the technique that consists in merging the immigrant Pareto front with the local one, and the result constitutes the new local Pareto archive. The number of emigrants can be expressed as a fixed or variable number of individuals, or as a percentage of individuals from the population or the Pareto archive. The choice of the value of such parameter is crucial. Indeed, if it is low the migration process will be less effcient as the islands will have the tendency to evolve in an independent way. Conversely, if the number of emigrants is high, the EAs will likely to converge to the same solutions (premature convergence).

Although most of works on parallel multi-objective metaheuristics are related to evolutionary algorithms, there are also proposals related to alternative methods, such us tabu search [2], simulated annealing [1] [9], ant colonies [12] and memetic algorithms [5].

1.2. Combining Metaheuristics with Exact Methods for MOP

Another recent popular issue is the cooperation between multi-objective metaheuristics and exact optimization algorithms. Some hybrid schemes mainly

aim at providing Pareto optimal sets in shorter time, while others primarily focus on getting better Pareto set approximations. In a multi-objective context, only few studies tackle this type of approaches.

Combining branch and bound with multi-objective metaheuristics: an investigation of several cooperative approaches combining multi-objective branch and bound [30] and multi-objective metaheuristics can be considered for MOPs [3]. A multi-objective metaheuristic which approximates the Pareto set of the problem [4], and a bi-objective branch and bound which has been designed to solve the bi-objective flow-shop scheduling problem [25].

Three hybrid schemes combining an exact algorithm with a multi-objective metaheuristic may be considered [3]:

- **Metaheuristic to generate an upper bound:** the first HRH hybrid exact scheme is a multi-objective exact algorithm (e.g. branch and bound) in which the Pareto set approximation is used to speedup the algorithm (Fig. 5). The Pareto set approximation is considered as a good upper bound approximation for the multi-objective exact algorithm. Hence, many nodes of the search tree can be pruned by the branch and bound algorithm. The time required to solve a given problem instance is smaller if the distance between the Pareto front approximation and the Pareto optimal front is small. If the distance is null, the exact algorithm will serve to prove the optimality of the Pareto set approximation. Even if this hybrid approach reduces the search time needed to find the Pareto optimal set, it does not allow to increase considerably the size of the solved instances.

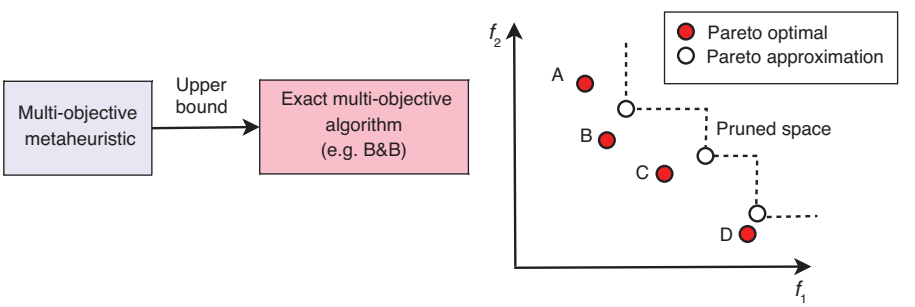

Figure 5. A HRH exact hybrid scheme in which a multi-objective metaheuristic generates an upper bound Pareto set to an exact multi-objective algorithm (e.g. branch and bound).

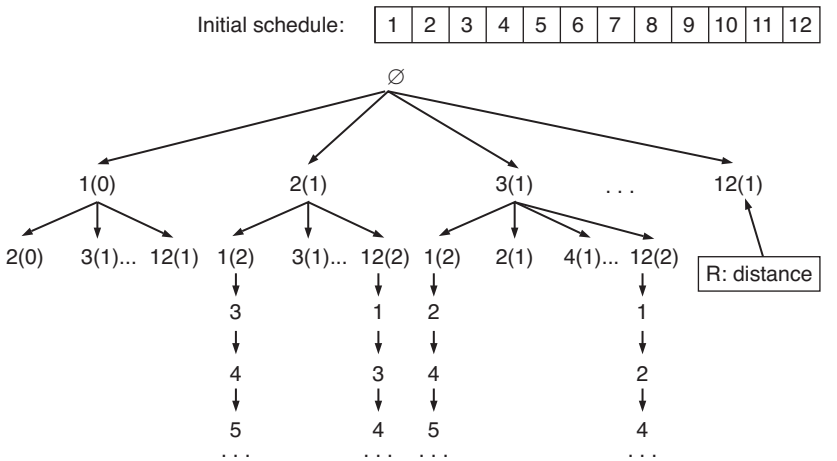

Figure 6. A hybrid heuristic scheme in which an exact algorithm explores very large neighborhoods of the multi-objective metaheuristic.

- **Exact algorithm to explore very large neighborhoods:** in this hybrid heuristic approach, the exact multi-objective algorithm is used to explore large neighborhoods of a Pareto solution. The main idea is to reduce the search space explored by the exact algorithm by pruning nodes when the solution in construction is too far from the initial Pareto solution.

  Let us consider a permutation based representation for the biobjective flow-shop scheduling problem, and an insertion neighborhood operator. The exact algorithm is allowed to explore the neighborhood of the initial Pareto solution in which the solutions are within a distance less or equal to δ_{max} (Fig. 6). The size of the insertion-based neighborhood is $\Theta(n^2)$, where n represents the number of jobs. Hence, the size of the search space explored by the exact algorithm is exponential and may be approximated by $\Theta(n^{2\delta\max})$. Then, the distance δ_{max} must be limited, especially for instances with large number of jobs.
- **Exact algorithm to solve subproblems:** in this hybrid heuristic approach, the exact multi-objective algorithm solve subproblems which are generated by the multi-objective metaheuristic. A given region of the decision space is explored by the exact algorithm. Figure 7 shows an example of such hybridization. Let us consider an initial Pareto

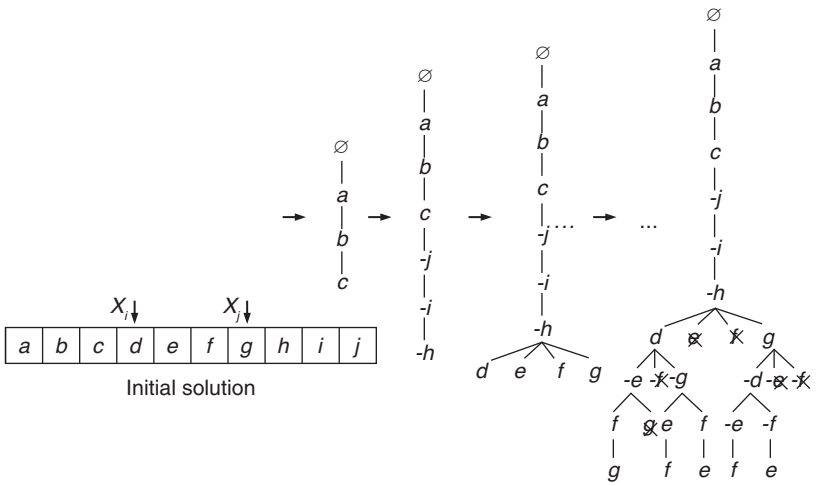

Figure 7. A hybrid heurisitic scheme in which an exact algorithm solves sub- problems generated by a multi-objective metaheuristic.

solution composed of 10 jobs (*a, b, ..., i, j*) which is obtained by the multi-objective metaheuristic. Subproblems of a given size (e.g. 4) are explored by the exact algorithm (e.g. the subproblem defined by the non-freezed jobs *d, e, f, g*). The first phase consists in placing the three first jobs at the beginning of the schedule. Moreover, the branch and bound algorithm places the three last jobs at the end of the schedule (a job *j* placed in queue is symbolized by *–j*). Then, the branch and bound multi-objective algorithm is applied on the remaining non-freezed jobs to generate all Pareto solutions in this subspace.

The main parameters which have to be defined for an efficient hybrid scheme are:

– **Partition sizes:** the cardinality of the Pareto set approximation obtained by a multi-objective metaheuristic varies according to the target MOP and instance. For the BOFSP problem, the size of the Pareto set approximation varies between several tens and two hundred solutions. Moreover, the size of partitions must be also limited according to the efficiency of the exact method at hand. For the BOFSP, it may be fixed to 25 jobs for 10 machines instances and

12 jobs for the 20 machines instances, so each exact method can be performed in several seconds or some minutes [3].

– **Number of partitions for each solution:** enough partitions of the complete schedule have to be considered to treat each job at least once by the exact method. Moreover, it is interesting to superpose consecutive partitions to allow several moves of a same job during optimization. Then, a job which is early scheduled could be translated at the end of the schedule by successive moves. On the other side, more partitions are considered, more important the computational time is. For instance, for the BOFSP, 8 partitions for the 50–jobs instances, 16 partitions for the 100–jobs and 32 partitions for the 200–jobs instances may be considered [3].

Combining branch and cut with multi-objective metaheuristics: this example investigates the solution of a multi-objective routing problem, namely the bi-objective covering tour problem (BOCTP), by means of a hybrid HRH strategy involving a multi-objective metaheuristic and a single-objective branch-and-cut algorithm. The BOCTP aims to determine a minimal length tour for a subset of nodes while also minimizing the greatest distance between the nodes of another set and the nearest visited node. The BOCTP can be formally described as follows (Fig. 9): let $G = (V \cup W, E)$ be an undirected graph, where $V \cup W$ is the vertex set, and $E = \{(v_i , v_j)/v_i , v_j \; V \cup W, i < j \}$ is the edge set. Vertex v_1 is a depot, V is the set of vertices that can be visited, $T \subseteq V$ is the set of vertices that must be visited ($v_1 \in T$), and W is the set of vertices that must be covered. A distance matrix $C = (c_{ij})$, satisfying triangle inequality, is defined for E. The BOCTP consists of defining a tour for a subset of V, which contains all the vertices from T, while at the same time optimizing the following two objectives: (i) the minimization of the tour length and (ii) the minimization of the cover. The cover of a solution is defined as the greatest distance between a node $w \in W$, and the nearest visited node $v \in V$.

The BOCTP problem has been extended from the mono-objective (CTP). The CTP problems consists in determining a minimum length tour for a subset of V that contains all the vertices from T, and which covers every vertex w from W that is covered by the tour (i.e. w lies within a distance c from a vertex of the tour, where c is a user defined parameter). A feasible solution for a small instance is provided in figure 9. One generic application of the CTP involves designing a tour in a network whose vertices represent points that can be visited, and from which the places that are not on the tour can be easily reached. In the bi-objective

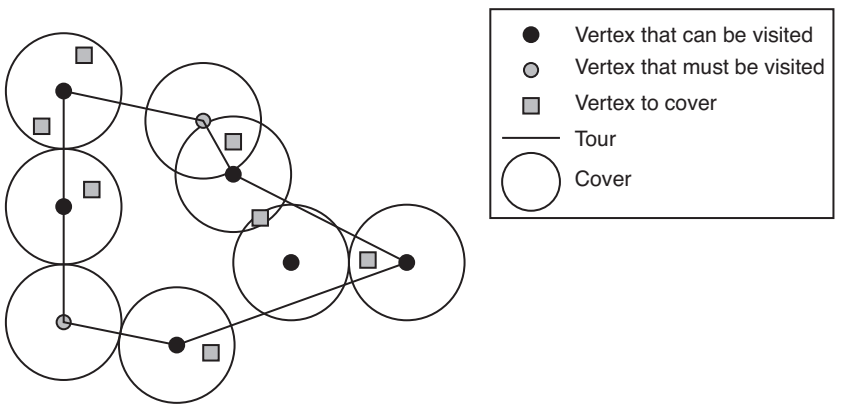

Figure 8. The covering tour problem: an example of a solution.

covering tour problem BOCTP, the constraint on the cover has been replaced by an objective in which the covering distance is minimized [24].

Let us consider a multi-objective metaheuristic to solve the BOCTP problem which approximates the Pareto set [24], and a branch and cut algorithm to solve the mono-objective CTP problem [14]. The branch and cut algorithm may be considered as a black box optimization tool whose inputs are a subset of V, the set W, and a cover, and whose output is the optimal tour for the CTP. The branch and cut algorithm first relaxes the integrality conditions on the variables and the connectivity constraints of the integer linear programming model. Integrality is then gradually restored by means of a branch and bound mechanism. Before initiating branching at any given node of the search tree, a search is conducted for violated constraints, including the initially relaxed connectivity constraints and several other families of valid constraints. Several classes of valid inequalities have been considered such as dominance constraints, covering constraints, sub-tour elimination constraints, and 2-matching inequalities [14].

In the hybrid approach, the multi-objective metaheuristic generates a Pareto set approximation, which is used to build subproblems; these subproblems are then solved using the branch and cut algorithm (Fig. 9). Subproblem construction is a key point of the cooperative design, given that prohibitive computational times result if the subsets of V are too large. By limiting their size and giving the branch and cut algorithm access to the information extracted from the Pareto set approximation, the method makes solving the subproblems relatively easy for

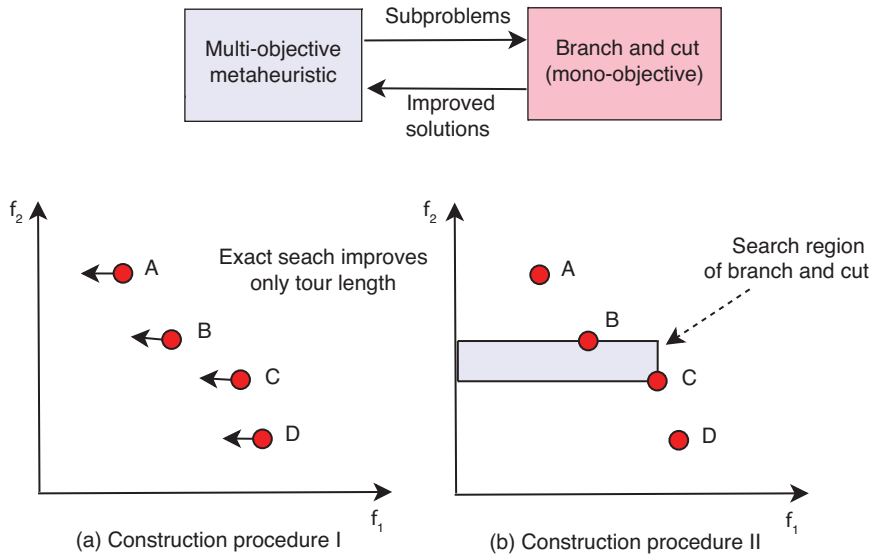

Figure 9. Combining a mono-objective branch and cut algorithm and a multiobjective metaheuristic to solve the bi-objective covering tour problem.

the branch-and-cut algorithm. Two procedures for building the subproblems can be considered [24]:

- **One objective improvement by an exact algorithm:** the main purpose of the first construction procedure is to improve the solutions found by the multi-objective metaheuristic in terms of the tour length objective without modifying the cover value. It accomplishes this goal by investigating the possibility that some elements of the set of visited vertices $\widetilde{V}$ can be replaced by sets of vertices $R \subseteq V \backslash \widetilde{V}$ so that the cover value $\widetilde{c}$ provided by the couple (v_t, v_c) remains unchanged (Fig. 9a). A vertex $v_k \in \widetilde{V}$ can be replaced by a set R if and only if: (i) No subset of R can replace v_k; (ii) No vertex from R can provide a better cover: $\forall v_i \in R, c_{tc} \, '' \, c_{ic}$; (iii) There must be a vertex from $\widetilde{V}$ or from R that can replace v_k for every vertex of W that can be covered by vk. Therefore, $\forall v_l \in W \backslash \{v_c\}$, such that $c_{kl} \, '' \, \widetilde{c}$, where the following condition must be true: $\exists v_n \in R \cup (\widetilde{V} \backslash \{v_k\}), c_{nl} \, '' \, \widetilde{c}$.

  Replacing a node of $\widetilde{V}$ by a subset R tends to become easier as the cardinality of R increases. However, in practice, condition (i) limits the

candidate subsets. The larger the R set, the higher the cost of the test. Certainly, if the size of the set used for the branch and cut algorithm is very large, the algorithm will require too much computational time. Therefore, in practice, the cardinality of R is limited to one or two elements.

For each solution s of the Pareto set approximation, a problem is built as follows. The set V_I of vertices that can be visited is created by the union of $\widetilde{V}$ and all subsets of V with a cardinality of 1 or 2 that can replace a vertex of $\widetilde{V}$. The set W of vertices that must be covered remains unchanged. Here, the parameter c is equal to the cover of s.

∞ **Region exploration by an exact algorithm:** in the first construction procedure it is unlikely that all the feasible covers corresponding to Pareto optimal solutions will be identified. These unidentified solutions must always be situated between two solutions of the approximation, although not always between the same two solutions. Thus, it is reasonable to assume that new Pareto optimal solutions may be discovered by focusing searches in the area of the objective space between two neighboring solutions. The second procedure aims to build sets of vertices in order to identify potentially Pareto optimal solutions whose cover values were not found by the multi-objective metaheuristic. Let A and B be two neighboring solutions in the approximation sets found by the evolutionary algorithm (i.e. there are no other solutions between A and B). A (resp. B) is a solution with a cover c_A (resp. c_B) which visits the vertices of the set V_A (resp. V_B). Assuming that $c_A < c_B$, the branch and cut algorithm can be executed on a set V_{II} , built according to both V_A and V_B , with the first cover $\widetilde{c}$ which is strictly smaller than c_B as a parameter (Fig. 9b). If $\widetilde{c}$ is equal to c_A , there is no need to execute the branch and cut algorithm.

It appears that neighboring solutions in the Pareto set have a large number of vertices in common. Thus, V_{II} contains V_A and V_B . This inclusion insures that the branch and cut algorithm will at least be able to find the solution A, or a solution with the same cover but a better tour length in cases for which the tour on V_A is not optimal. The following process is used to complete V_{II} : for every feasible cover c, so that $c_A < c < c_B$, vertices are added to V_{II} in order to obtain a subset of V_{II} with c as a cover. Algorithm 1.2 provides the procedure for constructing the set V_{II} .

Construction of the set V_{II} . $V_{II} = V_A \cup V_B$;
For all c so that $c_A < c < c_B$ **Do**
For $v_l \in W$ **Do**
$V_{II} = \cup V_{II} \{vk \in V \backslash V_{II} / ckl \, '' \, c\}$
End For

1.3. Combining Metaheuristics with Data Mining for MOP

Most of the classical combinations of metaheuristics with machine learning and data mining techniques (e.g. feature selection, classification, clustering, association rules) which have been applied to mono-objective optimization can be generalized to multi-objective optimization:

- ∞ Search operators (e.g. recombination operators in P-metaheuristics, neighborhoods in S-metaheuristics).
- ∞ Optimization models (e.g. approximation of the objectives functions, generation of sub-problems, new constraints).
- ∞ Parameter setting of the metaheuristics.

Search operators: integrating knowledge into search operators is the most popular scheme in this class of hybrids. For instance, in a P-metaheuristic (e.g. evolutionary algorithm), a set of decision rules describing the best and worst individuals of the current population may be extracted. The extracted rules are incorporated into the crossover operator of an evolutionary algorithm to generate solutions sharing the characteristics of non-dominated solutions and avoiding those of dominated solutions.

This principle can be applied to multi-objective optimization in the following way [21]: a set of rules that describes why some individuals dominate others (positive rules) and why some individuals are dominated by others (negative rules[3]) are extracted using the *C4.5 classifier*. Offsprings that match the positive rules and do not match the negative rules are generated. The obtained results indicate that those learnable evolution models allow to speedup the search and improve the quality of solutions.

Parameter setting: in a multi-objective metaheuristic, the efficiency of an operator may change during the execution of the algorithm: an operator may offer a better convergence at the beginning of the metaheuristic, but this convergence may be improved later with another operator. The success of an operator may also depend on the instance of the problem. This motivates the use of adaptive operator probabilities to automate the selection of efficient operators. The adaptation can be done by exploiting information gained, either

[3]Negative knowledge.

implicitly or explicitly, regarding the current ability of each operator to produce solutions of better quality [29]. Other methods adjust operator probabilities based on other criteria, such as the diversity of the population [10]. A classification of adaptation on the basis of the used mechanisms, and the level at which adaptation operates may be found in [16].

Adaptive mutation in multi-objective evolutionary algorithms: let us consider a multi-objective evolutionary algorithm in which the choice of the mutation operators is done dynamically during the search. The purpose is to use simultaneously several mutation operators during the EA, and to change automatically the probability selection of each operator according to its effiectiveness [5]. So the algorithm always uses more often the best operators than the others. Let us remark that a similar approach could be defined with other operators (e.g. crossover, neighborhoods, hybrid strategies).

Initially, the same probability is assigned to each mutation operator: $M\,u_1$, ... , $M\,u_k$. Those probabilities are equal to the same ratio $P_{Mui} = 1/(k * P_{Mu})$, where k is the number of mutation operators, and P_{Mu} is the global mutation probability. At each iteration, the probabilities associated to the mutation operators are updated according to their average progress. To compute the progress of the operators, each mutation M_{ui} applied to the individual I is associated with a progress value:

$$\text{II}(I_{Mui}) = \begin{cases} 1 \text{ if } I \text{ is dominated by } I_{Mui} \\ 0 \text{ if } I \text{ dominates } I_{Mui} \\ \frac{1}{2} \text{otherwise (non comparable solutions)} \end{cases}$$

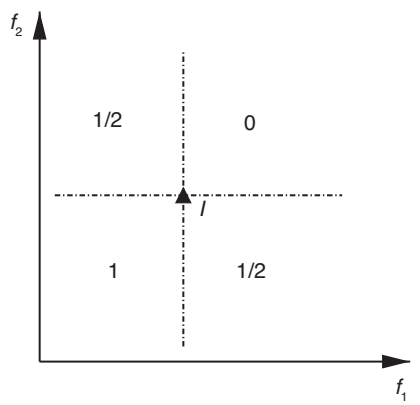

Figure 10. Progress value of $\text{II}(I_{Mui})$ for mutation operators.

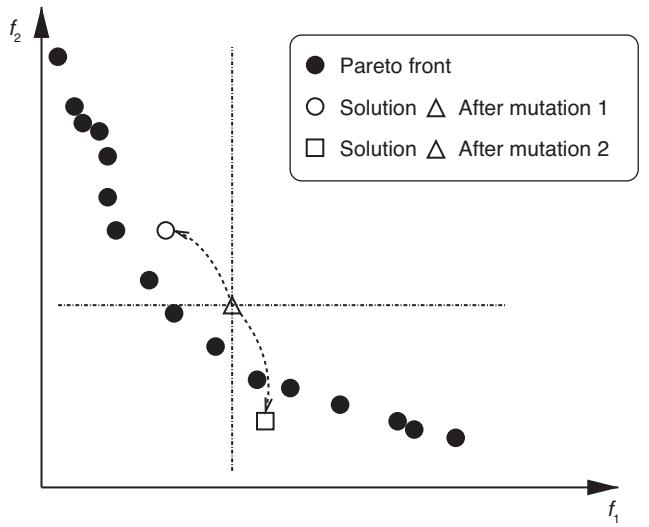

Figure 11. Evaluation of the quality of the mutation operators.

where I_{Mui} is the solution after mutation (Fig. 10).

At the end of each generation of the EA, an average progress *Progress* (Mu_i) is assigned to each operator M_{ui}. Its value is the average progress of II(IM_{ui}) computed with each solution modified by the mutation M_{ui} :

$$Progress\,(Mu_i) = \frac{\sum \Pi(I_{Mu_i})}{\|Mu_i\|}$$

where $\|M_{ui}\|$ is the number of applications of the mutation M_{ui} on the population. The new selection probabilities are computed proportionally to these values:

$$P_{Mu_i} = \frac{Progress(Mu_i)}{\sum_{j=1}^{k} Progress(Mu_j)} \times (1 - k \times \delta) + \delta$$

where δ is the minimal selection probability value of the operators.

This approach of progress computation compares two solutions with their dominance relation. However, a comparison only between I and I_{Mui} is not sufficient. Firstly, if the two individuals I and I_{Mui} are non comparable, the quality of the mutation cannot be evaluated. For instance, in figure 11, the progress II of the two mutation operators applied on the solution Δ is the same (1/2). However,

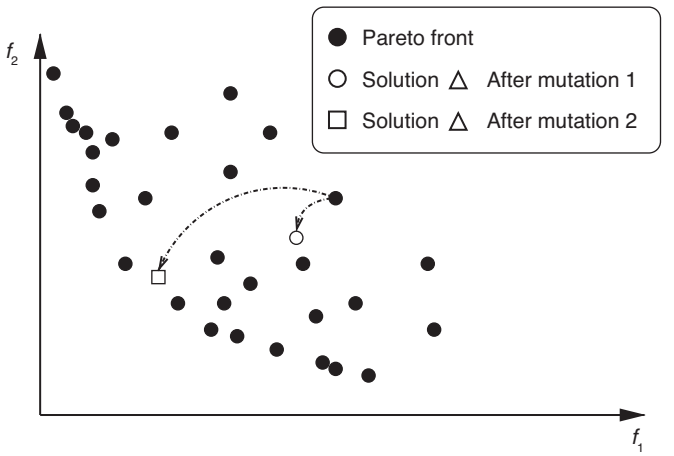

Figure 12. Computing the progress realized by diferent mutation operators.

the observation of the whole Pareto front shows that the second mutation operator performs better since the generated solution by the second mutation operator is Pareto optimal whereas the solution generated by the first is not.

Secondly, if the generated individual dominates the initial individual, the progress realized cannot be measured with precision. For instance, in figure 12, the progress II of the two mutation operators applied on the solution Δ. is the same (1). However, the observation of the whole population shows that the second mutation operator performs much better than the first one.

These problems can be tackled in the case of evolutionary algorithms using selection by ranking. The progress value can be replaced by:

$$\Pi(I_{Mu_i}) = \left(\frac{R_k I}{R_k I_{Mu_i}} \right)^{\beta}$$

where $R_k\, I_{Mui}$ is the rank of the solution after mutation, $R_k\, I$ is the rank of the solution before mutation, and β is how much the progress made by mutation operators is encouraged (e.g. $\beta = 2$).

The evaluation of the progress of the mutation operators can be still improved by supporting the progresses realized on good solutions. In fact, these progresses are generally more interesting for the front progression than progresses made on bad solutions (Fig. 13). So an elitist factor $E_f\, I_{Mui}$ has been has been introduced into the last progress indicator to favor progresses made on good solutions:

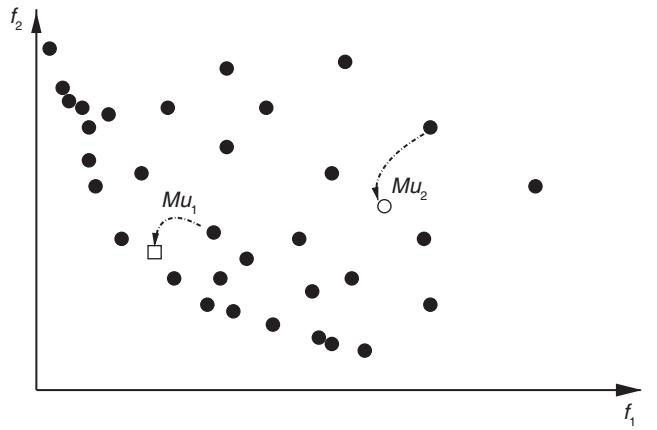

Figure 13. Progress realized by mutation operators on good quality solutions.

$$\Pi(I_{Mu_i}) = E_f I_{Mu_i} \times \left(\frac{R_k I_{Mu_i}}{R_k I} \right)^{\beta}$$

with $E_f I_{Mui} = (R_k I_{Mui})^{-1}$. Then, the average progress of a mutation Mu_i is defined as follows:

$$Progress(Mu_i) = \frac{\sum \Pi(I_{Mu_i})}{\sum E_f I_{Mu_i}}$$

Some hybrid schemes are specific to multi-objective metaheuristics such as introducing data mining tasks in the search component dealing with elitism.

Clustering archives in multi-objective metaheuristics: a classical approach using data mining approaches in the population management of multi-objective metaheuristics is the application of clustering algorithms on the archive. The objective is to produce a set of well diversified representatives Pareto solu- tions in a bounded archive. An archive is often used to store Pareto solutions and the clustering is then performed to avoid a bias towards a certain region of the search space and to reduce the number of Pareto solutions. Such a bias would lead to an unbalanced distribution of the Pareto solutions. For instance, a hierarchical clustering can be applied using the average linkage method [31].

2. CONCLUSIONS AND PERSPECTIVES

The efficient solving of complex MOPs must involve ideas from different paradigms: metaheuristics, mathematical programming, constraint programming, machine learning, graph theory, parallel and distributed computing, and so on. Pure metaheuristics are not generally well suited to search in high-dimensional and complex landscapes. Hybrid metaheuristics represent actually the most efficient algorithms for many classical and real-life difficult MOPs. This is proven by the huge number of efficient hybrid metaheuristics proposed to solve a large variety of problems.

A unified taxonomy has been developed to describe in terms of design and implementation the different hybridization schemes of metaheuristics for MOPs.

The main drawback of hybridization is the introduction of new parameters which define the hybrid scheme. The setting of those parameters is non trivial. A crucial question that has to be addressed in the future is an aid for the efficient design of hybrid metaheuristics, in which the automatic setting of parameters must be investigated.

It will be also interesting to deeply explore parallel models for hybrid methods. Parallel schemes ideally provide novel ways to design and implement hybrid algorithms by providing parallel models of the algorithms. Hence, instead of merely parallelizing and finely tuning a sequential hybrid algorithm which has limited capabilities to be parallelized, teamwork hybrid schemes are inherently suited to parallel environments.

REFERENCES

[1] D. K. Agrafiotis. Multiobjective optimization of combinatorial libraries. Technical report, IBM J. Res. and Dev., 2001.

[2] A. Al-Yamani, S. Sait, and H. Youssef. Parallelizing tabu search on a cluster of heterogeneous workstations. *Journal of heuristics*, 8(3): 277–304, 2002.

[3] M. Basseur, J. Lemesre, C. Dhaenens, and E.-G. Talbi. Cooperation between branch and bound and evolutionary approaches to solve a biobjective flow shop problem. In *Worshop on Experimental Algorithms (WEA'04)*, pages 72–86. LNCS Vol.3059, Springer, 2004.

[4] M. Basseur, F. Seynhaeve, and E-G. Talbi. Design of multi-objective evolutionary algorithms: Application to the flow-shop scheduling problem. In *Congress on Evolutionary Computation CEC'02*, pages 1151–1156, Honolulu, Hawaii, USA, May 2002.

[5] 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.

[6] M. Basseur, F. Seynhaeve, and E-G. Talbi. Path relinking in Pareto multi-objective genetic algorithms. In C. A. Coello Coello, A. H. Aguirre, and E. Zitzler, editors,

Evolutionary Multi-Criterion Optimization, EMO'2005, volume 3410 of *Lecture Notes in Computer Science*, pages 120–134, Gua-najuato, Mexico, 2005. Springer-Verlag.

[7] R. P. Beausoleil. Mutiple criteria scatter search. In *4th Metaheuristics International Conference (MIC'01))*, pages 539–544, Porto, Portugal, 2001.

[8] E. K. Burke, J. D. Landa Silva, and E. Soubeiga. Hyperheuristic approaches for multiobjective optimisation. In *5th Metaheuristics International Conference (MIC'2003)*, Kyoto, Japan, August 2003.

[9] C. S. Chang and J. S. Huang. Optimal multiobjective SVC planning for voltage stability enhancement. *IEE Proceedings on Generation, Transmission and Distribution*, 145(2):203–209, 1998.

[10] J. Coyne and R. Paton. Genetic algorithms and directed adaptation. In *Workshop on Evolutionary Computing*, pages 103–114, Leeds, UK, 1994. LNCS Vol.865, Springer.

[11] K. Deb and T. Goel. A hybrid multi-objective evolutionary approach to engineering shape design. In E. Zitzler, K. Deb, L. Thiele, C. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 385–399, Zurich, Switzerland, 2001.

[12] P. Delisle, M. Kra jecki, M. Gravel, and C. Gagné. Parallel implementation of an ant colony optimization metaheuristic with OpenMP. In *3rd European Workshop on OpenMP (EWOMP'01)*, pages 8–12, 2001.

[13] M. Gen and L. Lin. Multiobjective hybrid genetic algorithm for bicriteria network design problem. In *The 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, pages 73–82, Cairns, Australia, December 2004.

[14] M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45:568576, 1997.

[15] I. E. Golovkin, S. J. Louis, and R. C. Mancini. Parallel implementation of niched Pareto genetic algorithm code for x-ray plasma spectroscopy. In *Congress on Evolutionary Computation (CEC'02)*, pages 1820–1824, 2002.

[16] R. Hinterding, Z. Michalewicz, and A-E. Eiben. Adaptation in evolutionary computation: A survey. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 65–69, Indianapolis, USA, April 1997.

[17] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 28(3):392–403, Aug 1998.

[18] A. Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. Technical Report RA-014/98, Institute of Computing Science, Poznan University of Technology, 1998.

[19] J. Jaszkiewicz. Path relinking for multiple objective combinatorial optimization: TSP case study. In *The 16th Mini-EURO Conference and 10th Meeting of EWGT (Euro Working Group Transportation)*, 2005.

[20] B. R. Jones, W. A. Crossley, and A. S. Lyrintzis. Aerodynamic and aeroacoustic optimization of airfoils via parallel genetic algorithm. *Journal of aircraft*, 37(6):1088–1098, 2000.

[21] L. Jourdan, D. Corne, D. A. Savic, and G. A. Walters. Preliminary investigation of the learnable evolution model for faster/better multiobjective water systems design. In *Int. Conf. on Evolutionary Multi-criterion Optimization EMO'05*, pages 841–855. LNCS 3410, Springer, 2005.

[22] N. Jozefowiez. *Modélisation et résolution approchée de problèmes de tournées multi-objectif*. PhD thesis, University of Lille, Lille, France, 2004.

[23] N. Jozefowiez, F. Semet, and E-G. Talbi. Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In J. Guervos, P. Adamidis, H-G. Beyer, J-L. Fernández-Villacanas, and H-P. Schwefel, editors, *Parallel Problem Solving from Nature (PPSN VII)*, number 2439 in Lecture Notes in Computer Science, pages 271–280, Granada, Spain, 2002. Springer-Verlag.

[24] N. Jozefowiez, F. Semet, and E-G. Talbi. The bi-objective covering tour problem. *Computers and Operations Research*, 34:1929–1942, 2007.

[25] J. Lemesre, C. Dhaenens, and E-G. Talbi. An exact parallel method for a bi-objective permutation flowshop problem. *European Journal of Operational Research (EJOR)*, 177(3):1641–1655, 2007.

[26] H. Meunier, E.-G. Talbi, and P. Reininger. A multiobjective genetic algorithm for radio network optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation (CE'00)*, pages 317–324, La Jolla, CA, USA, 2000. IEEE Press.

[27] J. Rowe, K. Vinsen, and N. Marvin. Parallel GAs for multiobjective func- tions. In *Proc. of the 2nd Nordic Workshop on Genetic Algorithms and Their Applications* (2NWGA), pages 61–70, 1996.

[28] E-G. Talbi, M. Rahoual, M-H. Mabed, and C. Dhaenens. A hybrid evolutionary approach for multicriteria optimization problems: application to the flow shop. In E. Zitzler et al., editor, *First International Conference on Evolutionary Multi-Criterion Optimization EMO'01, LNCS No.1993*, pages 416–428, Zurich, Switzerland, 2001. Springer Verlag.

[29] A. Tuson and P. Ross. Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2):161–184, 1998.

[30] M. Visée, J. Teghem, M. Pirlot, and E. L. Ulungu. Two-phases method and branch and bound procedures to solve knapsack problem. *Journal of Global Optimization*, 12:139–155, 1998.

[31] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, 1999.