MO-Greedy: an extended beam-search approach for solving a multi-criteria scheduling problem on heterogeneous machines

Louis-Claude Canon
Ensimag, INRIA Rhône-Alpes

Emmanuel
INRIA Bordeaux Sud-Ouest, LaBRI

***Abstract*—Optimization problems can often be tackled with respect to several objectives. In such cases, there can be several incomparable Pareto-optimal solutions. Computing or approximating such solutions is a major challenge in algorithm design. Here, we show how to use an extended beam-search technique to solve a multi-criteria scheduling problem for heterogeneous machines. This method, called MO-GREEDY (for Multi-Objective greedy), allows the design of a multi-objective algorithm when a single-objective greedy one is known. We show that we can generate, in a single execution, a Pareto front optimized with respect to the preferences specified by the decision maker. We compare our approach to other heuristics and an approximation algorithm and show that the obtained front is, on average, better with our method.**

I. INTRODUCTION

A greedy algorithm is an algorithm in which made decisions are never taken back (see [6, Chap. 16]). Such an approach has proved to be successful for designing optimal algorithms (*e.g.*, Euclid's algorithm, Huffman coding); approximation algorithms (*e.g.*, *best fit decreasing* for the 2D-bin-packing problem) or efficient heuristics (*e.g.*, Heterogeneous Earliest Finish Time [12] for a scheduling problem on heterogeneous machines).

In multi-objective optimization, there can be several incomparable Pareto-optimal solutions. Computing or approximating such solutions (called the Pareto front) is a major challenge in algorithm design. There exists several methods to compute or approximate a Pareto front such as genetic algorithms, approximation algorithms, aggregation based approaches or heuristics. However, there does not exists dedicated frameworks that generalize the greedy approach for the multi-objective case.

Beam-search [15] is a heuristic that explores a graph by keeping only most promising nodes and discarding other nodes. It is similar to a depth-first search with the difference that only a subset of the nodes is kept for further exploration.

In this article, we propose to use the beam-search as a generalization of the greedy approach for multi-criteria problem. We call this approach MO-GREEDY for Multi-Objective greedy. More precisely, we present how to transform a greedy single-objective strategy into a multi-objective one. The resulting strategy constructs, in a greedy manner, multiple solutions. The generated set of solutions is intended to achieve good quality with respect to the preferences defined by the decision maker.

We have applied our method to a scheduling problem and we use the hypervolume [18] as a set preference relation for assessing the quality of Pareto set approximations. We have conducted a large set of experiments with two aims. The first is to characterize the good values of the parameters of the strategy. The second lies in comparing the MO-GREEDY strategy to other efficient heuristics or optimal methods. Experimental results show that our approach leads to good solutions outperforming other heuristics such as aggregation or approximation algorithms.

II. MODEL AND PROBLEM SETTING

Let us consider a problem for which a solution π can be constructed incrementally by performing a succession of g greedy decisions. At step i, the decision d_i is selected among a set of possible decisions denoted as D_i. Let $\pi_i = (d_1, \ldots, d_i)$ be the partial solution obtained after the i-th step. Then, the final solution can be defined as a succession of g decisions $\pi = \pi_g$.

Let o be the number of objectives of a multi-objective optimization problem. Let $f_i(\pi)$ be the value of the i-th objective of solution π. Then, we define the objective vector $f(\pi) = (f_1(\pi), \ldots, f_o(\pi))$ to be the vector of objective values of solution π.

Without loss of generality, each of the o objectives has to be minimized. There may not exist any solution that is optimal for each objective and two solutions might be incomparable (none of them dominates the other for each objective). The Pareto-dominance relation formalizes the concept of optimality in the case of multiple objectives. Solution π is said to weakly Pareto-dominates π' (*i.e.*, $\pi \preceq \pi'$) if $\forall i \in [1..o], f_i(\pi) \leq f_i(\pi')$. Moreover, solution π Pareto-dominates π' if $\pi \preceq \pi'$ and if $\pi \neq \pi'$. Solution π is Pareto-optimal if no other solution Pareto-dominates it. The Pareto front contains all Pareto-optimal solutions.

Our study consists in generating a set of solutions for a multi-objective problem by generalizing the greedy principle (we never take back any decision previously made). The goal is to build a set of solutions approximating the Pareto front, called *Pareto set approximation*.

Pareto set approximations generated by any strategy might again be incomparable. Assessing the quality of a given Pareto set approximation has raised a recent focus [3], [18]. We use the generic concept of *set preference relation* that has been formalized by Zitzler et al. in [18]. Intuitively, a set preference relation says whether a Pareto set approximation is better than another one. The most basic relation is the weak Pareto-dominance relation $\preccurlyeq_{\text{par}}$ which is defined as $A \preccurlyeq_{\text{par}} B \Leftrightarrow (\forall b \in B : (\exists a \in A : a \preceq b))$. However, in most of the cases, approximations are incomparable using that relation. It has then to be refined by the decision maker in order to convey informal preferences that are specific to the decision context (see [19] for a survey of indicators producing such relations with their advantages and drawbacks.).

Two concepts are usually used to denote the quality of an approximation set: the proximity and the diversity. The first indicates the distance between an approximation and the Pareto-optimal front, while the second concerns the distribution of the points in the objective space. These two concepts are often antagonist and are difficult to consider concurrently. The advantage of using a preference set relation is that the importance given to each concept is determined by how the decision maker refined the weak Pareto-dominance relation.

III. The MO-Greedy Method

As introduced in previous sections, we focus on finding optimized solutions that are produced through a succession of decisions and that possess each several objective values.

A. Formal Description

The MO-Greedy approach is a strategy that builds, through a succession of greedy decisions, several solutions that form a Pareto set approximation. The construction of these solutions can be seen as a tree search where nodes are partial solutions, edges are greedy decisions and leafs are final solutions. As the tree can be exponentially large, we cannot systematically explore it exhaustively. The beam-search strategy aims at exploring such a tree by keeping only a set of candidate nodes among the most promising ones. MO-Greedy is a formal extension of this strategy for our problem where, at each step, we maintain a set of partial solutions that leads eventually to a Pareto set approximation.

The difference between the greedy and the MO-Greedy methods is shown in Figure 1. In the greedy strategy, only one partial solution is kept at each step. In the MO-Greedy strategy, as we want a set of final solutions that approaches the Pareto front, we maintain a pool of partial solutions Π. Algorithm 1 formally describes the MO-Greedy approach and works as follows. At each step and for each partial solution $\pi \in \Pi$, we build l elements[1] π' by tacking greedy decisions(line 7). Such greedy decisions can be directly inspired from well-known single-objective greedy strategy as we will show for our scheduling problem. Hence, this method generalizes a single-objective greedy algorithm. If an element is a final solution (a leaf node), then we put it in the set of final solutions Π_{end} (line 10). Otherwise, it is a partial solution and we put it in the set of partial solutions Π_{next} (line 12). As in the beam-search strategy and in order to limit furthermore the exploration (for memory and time constraints), only k partial solutions are kept in set Π_{next} (line 13). Finally, these newly incremented and selected partial solutions are used as the partial solution set for the next step (line 4).

Algorithm 1: The MO-Greedy method

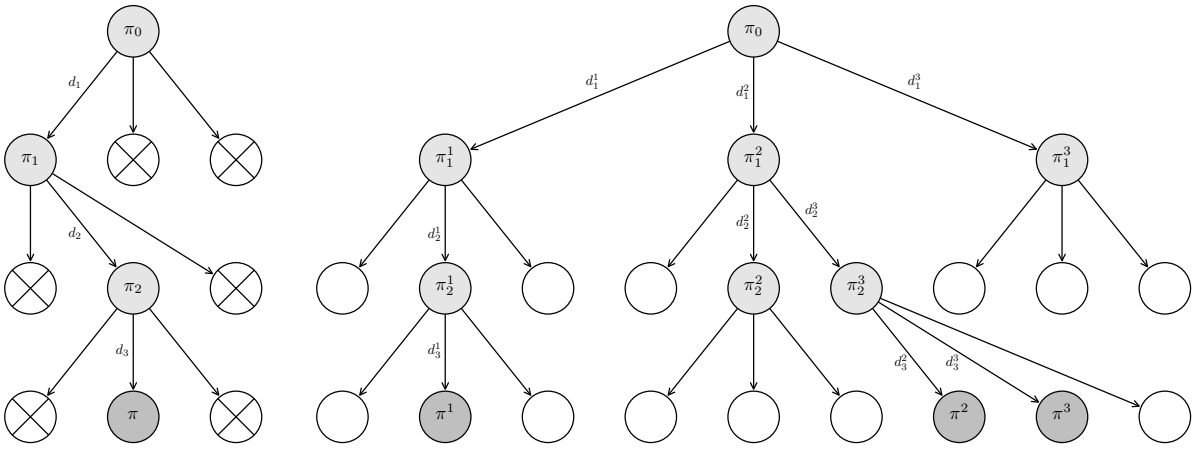
Input: l `// The maximum number of solutions generated from a partial solution`
Input: k `// The maximum number of partial solutions`
Input: D `// The set of possible greedy decisions`
Input: $\preccurlyeq$ `// The set preference relation`
Result: Π_{end} `// The set of final solutions`

1 $\Pi_{\text{end}} \leftarrow \emptyset$
2 $\pi_0 \leftarrow \emptyset$ `// An empty solution to start with`
3 $\Pi \leftarrow \{\pi_0\}$ `// The set of partial solutions`
4 **while** $\Pi \neq \emptyset$ **do**
5 **foreach** $\pi \in \Pi$ **do**
6 $\Pi_{\text{next}} \leftarrow \emptyset$
7 **foreach** $d \in$ *decision*$(D, \pi, \preccurlyeq, l)$ **do**
8 $\pi' \leftarrow \pi \cup \{d\}$
9 **if** π' *is a final solution* **then**
10 $\Pi_{\text{end}} \leftarrow \Pi_{\text{end}} \cup \pi'$
11 **else**
12 $\Pi_{\text{next}} \leftarrow \Pi_{\text{next}} \cup \pi'$
13 $\Pi \leftarrow$ *select*$(\Pi_{\text{next}}, \preccurlyeq, k)$

B. Selecting Elements and Comparing Fronts

At two points in Algorithm 1, a subset of elements is selected: in the *decision* function, we generate at most l

[1]We used the term *element* to encompass both final and partial solutions.



(a) Example of a greedy approach.

(b) Example of a MO-GREEDY execution (all decisions are considered). Where $l = k = 3$. The dark gray solutions represent the final solutions that constitute the output of the method.

Figure 1: Examples of two tree decision traversals.

new solutions from a partial solution; and in the *select* function, we keep k elements among all the elements of the partial solution set. Formally, the problem is the same in both cases: we have a front A of n elements and we want to build a front A' that contains only $m < n$ elements.

Selecting a subset of elements must be done carefully because elements that are not selected are destroyed and the subtree starting from this element will not be explored.

The selection process is done with a *set preference relation* ($\preccurlyeq$), a parameter of the MO-GREEDY method. Basically, a set preference relation says whether a Pareto set approximation is better than another one. Among all the possible subsets of A of size m, we want to select the one of best quality with respect to the relation $\preccurlyeq$.

To achieve this, one could generate all the possible subsets of size m and retains the one that is the best according to the relation $\preccurlyeq$. However, these comparisons between all possible subsets would be intractable.

Instead, we use the heuristics proposed in [18, Algorithms 4 and 5]. For both heuristics, set A is iteratively reduced by one element at a time. The element removed from the set is the one that contributes the least to the relation $\preccurlyeq$. The difference between Algorithm 4 and Algorithm 5 in [18], is that for determining the contribution of an element, a comparison between fronts is performed in Algorithm 4, while Algorithm 5 optimizes the choice when the relation $\preccurlyeq$ is based on an unary indicator by computing the contribution of each element directly. Although there is no guarantee that these algorithms lead to the best subset of elements, they are effective in practice.

As stated in Section II, the decision maker has to express what is a good Pareto set approximation (*i.e.*, he has to define the set preference relation $\preccurlyeq$). He can either customized an existing one or propose a new relation.

Note that the same relation $\preccurlyeq$ is used in both functions. However, this is not a requirement.

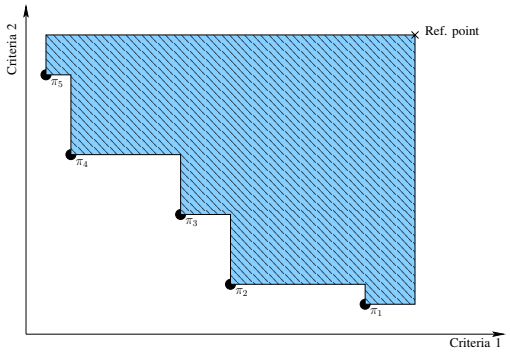


Figure 2: With two criteria, the hypervolume of the front with respect to the reference point is the colored zone.

C. Implementation Details

In our implementations, we are mainly using a set preference relation denoted $\preccurlyeq_H$ that is based on the *hypervolume*.

The hypervolume indicator [19] measures the volume (or the set of points) of the objective space dominated by a given front and that dominates a reference point (see Figure 2). Intuitively, it corresponds to the extend of a front from the reference point. Whether the considered criteria have to be minimized or maximized, the hypervolume needs to be maximized by choosing a relevant reference point. This indicator favors the proximity of an approximation set to the Pareto-optimal front over its diversity.

Algorithm 2: The MO-GREEDY strategy applied to the scheduling problem

Input: l // The maximum number of schedules generated from a partial schedule
Input: k // The maximum number of partial schedules
Input: G // The input task graph
Input: P // The processor set of size m
Input: $\preccurlyeq_H$ // The set preference relation
Result: Π_{end} // The set of final schedules

$D \leftarrow$ set of task sorted according to the HSA order
$\Pi_{\text{end}} \leftarrow \emptyset$
$\pi_0 \leftarrow \emptyset$ // An empty schedule to start with
$\Pi \leftarrow \{\pi_0\}$ // The set of partial schedules
while $\Pi \neq \emptyset$ **do**
  $t \leftarrow$ next task in set D
  foreach $\pi \in \Pi$ **do**
    $\Pi_{\text{next}} \leftarrow \emptyset$
    foreach *processor* $p_i \in P$ **do**
      $\pi^i \leftarrow \pi \cup$ schedule task t on processor p_i
    if $l > m$ **then**
      Keep the l best π^i ($i \in [1, p]$) according to $\preccurlyeq_H$ and the method described in Sec. III-B and III-C using the *failure probability* vs. *shifted makespan* objective space
    if t *was the last task to schedule* **then**
      Put the l kept schedules π^i in set Π_{end}
    else
      Put the l kept schedules π^i in set Π_{next}
  if $|\Pi_{next}| > k$ **then**
    Keep the k best schedule of Π_{next} according to $\preccurlyeq_H$ and the method described in Sec. III-B and III-C using the *failure probability* vs. *shifted makespan* objective space
    Put these schedule in Π
  else
    $\Pi \leftarrow \Pi_{\text{next}}$

The way two Pareto set approximations are compared for a preference relation is described in [18]. It consists of two steps. First, the elements are partitioned into sets of incomparable elements: all elements that are not dominated by any other one are selected in the initial set. These selected elements are then removed from the initial set and put into a new partition. This operation is run successively until no more element remains in the initial set. The second step consists in measuring the quality of each partition with the indicator corresponding to the relation. The output is a vector of hypervolume values. A Pareto set approximation A is better than B if the vector of values obtained for the approximation A is lexicographically better than the one for B.

We recall that the MO-GREEDY method is not limited to this set preference relation.

IV. THE SCHEDULING PROBLEM: MAKESPAN VS. RELIABILITY

The authors of [8] presented a bi-objective scheduling problem for related machines. The goal is to schedule an application modeled by a task graph that is directed and acyclic (a DAG) on a set of processors that have different speeds and different failure rates. Two objectives have to be minimized: the makespan and the probability of failure of the whole application. The authors have shown that both criteria are contradictory. The following problem definition is taken from the [8].

A. Problem Definition

We model the application by a task graph: let $G = (T, E)$ be a Directed Acyclic Graph (DAG) with T a set of $n = |T|$ vertices (that represents tasks) and E a set of edges that represents precedence constraints among the tasks. Each task $t_i \in T$ is given a number of instructions o_i. Each edge is associated to the time necessary to

send data from one task to another one if they are not executed on the same processor. We are given a set P of m processors and processor $p_j \in P$ is associated with two values: τ_j the *unitary instruction execution time*, *i.e.*, the time to perform one instruction, and λ_j the *failure rate*. We assume that, during the execution of the DAG, the failure rate is constant. This means that the failure model follows an exponential law. Hence, each task t_i executed on processor p_j will last $o_i \times \tau_j$ and the probability that this task finishes (correctly) its execution is given by $e^{-o_i \times \tau_j \times \lambda_j}$.

A schedule is an assignment of the tasks to the processors such that, at most one task is executed at a time on any given processor and such that the precedence constraints are respected. We call $\text{proc}(i)$ the processor assigned to task t_i and C_i its completion time. The makespan is $C_{\max} = \max\{C_i\}$.

The reliability of a schedule is the probability that it finishes correctly and is given by the probability that all the processors are functional during the execution of all their assigned tasks, *i.e.*, $p_{\text{succ}} = e^{-\sum_{j=1}^{m} C^j_{\max} \lambda_j}$, where $C^j_{\max} = \max_{i|\text{proc}(i)=j}\{C_i\}$ is the completion time of the last task executed on processor p_j. Finally, the probability of failure of a schedule is given by $p_{\text{fail}} = 1 - p_{\text{succ}}$.

The problem we address is to find a schedule such that p_{fail} and $C_{\max}$ are minimized given a task graph and a set of heterogeneous processors.

B. Adaptation to MO-GREEDY

The adaptation of the MO-GREEDY strategy to our scheduling problem is described in Algorithm 2. It is depicted to follow as closely as possible the structure of the general MO-GREEDY method (Alg. 1). However, some obvious optimization can easily be performed to optimize the execution (such as the termination case).

For this adaptation, we use a variant of a greedy scheduling algorithm where tasks are considered in a total order that respects the partial order given by the DAG. In our implementation (line 1), we use the HSA order [11] which is the sum of the dynamic top level and the static bottom level (the dynamic bottom top level is recomputed each time a predecessor of a task is scheduled). The partial solutions set consists in schedules for which the same tasks are mapped in the same order to different processors. At each step of the algorithm and for each partial solution, we map the next task taken in the HSA order (line 6) to all of the m processors (line 10).

We face an interesting problem when comparing two different solutions in the objective space (makespan and reliability). Indeed, it happens that mapping a task to a given processor does not increase the makespan but only changes the reliability (see Figure 3a). In order to better compare such schedules (different mapping, different reliability but same makespan), a first idea is to use only the reliability objective. However, this is a mistake as, in general, both partial solutions will lead to different schedules and hence are incomparable. In order to obtain an objective space in which these solutions are comparable, we map them into the following space: the unchanged reliability and the *shifted makespan* (see Figure 3b). The shifted makespan m'_i of a partial schedule π^i (each partial solution is ranked first by increasing makespan and then by decreasing failure probability) is computed as follows $m'_i = m_i + \sum_{j=1}^{i-1} \delta_j$ for $i > 1$ ($m'_1 = m_1$), where m_i is the makespan of the partial solution and δ_j is the difference between the finish time of the new allocated task in schedule π^{j+1} and π^j. Note that the previously ranking of the solution is such that all δ_j are positive.

Hence, when m is greater than l (the number of new solutions per partial solution), we apply the following decision strategy. We build the m partial solutions and compare them using the *failure probability* vs. *shifted makespan* objective space to compare them (line 12).

Therefore if l equals 2, we use the processors that minimizes the finish time of the task (degenerating to the classic EFT policy) and the one that minimizes the probability of failure of the schedule.

When the total number of partial solutions is greater than k (line 17), we use the same strategy as for the decision phase to select k solutions, thus reducing the size of this set (line 18). Then, this set constitutes the next solution set to be expanded (line 19).

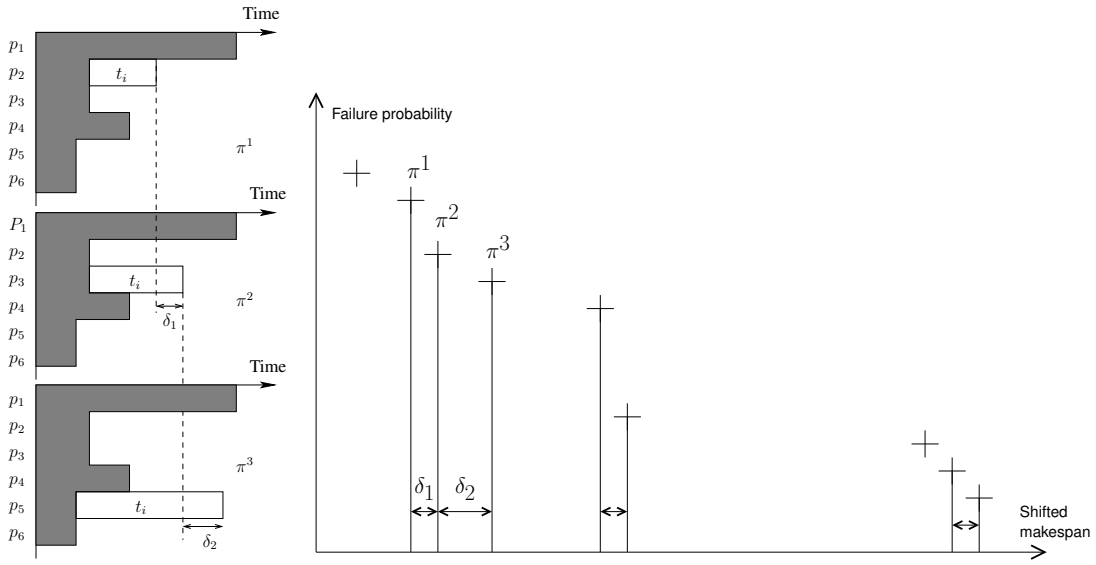
The algorithms continues until we map the last task. In this case, the partial schedules are put in Π_{end} (line 14) and the partial solution set Π_{next} remains empty.

V. EMPIRICAL VALIDATION

We validate our method empirically by studying its performances when applied to the scheduling problem described above. The code and the data are available on http://moais.imag.fr/membres/louis-claude.canon/MO-Greedy.tar.gz.

A. Instances

Each scheduling instance comprises a task graph and a platform. Task graphs are obtained from 4 algorithms: the Cholesky decomposition (referred to as *chol*); the Gaussian elimination, *ge*; the Gaussian elimination followed by a backward substitution, *gosser*; and, the Strassen matrix multiplication, *strassen*. An example task graph is given for each of these algorithms in Figure 4. The number of tasks varies from 1 to 100 (by increment of 1) for the first 3 instances. For *strassen*, the structure of the task graph remains unchanged for



(a) Three Gantt charts of the different partial schedules with the same makespan

(b) Comparison of the partial schedules in the *failure probability* vs. *shifted makespan* objective space

Figure 3: Comparing three partial solutions π^1, π^2 and π^3. Each has the same makespan (they come from the same partial solution π_{i-1} and task t_i does not increase the makespan) but they have different reliabilities: the probability of failure of π^1 is greater than for π^2, which is greater than for π^3. Each of these partial solutions should be considered as incomparable. For instance, mapping the next task t_{i+1} on processor p_5 would lead to a completely different trade-off depending on which of the three partial solutions is used. To take that into account, we shift the makespan objective to reflect the different possible trade-offs at this stage of the algorithm.

most inputs. Therefore, we select 2, 30, 128, 129, 180, 256, 257, 330, 512, 513, 700, 1024, 1025, 1300 and 2000 as the sizes of the multiplied matrices. We use 3 sets of processors. Each set contains 20 heterogeneous and unreliable processors. Speeds and the inverses of the failure rates are randomly generated according to a uniform distribution. There is no latency, the network is homogeneous and the topology is supposed to be complete. On the 3 generated platforms, unitary instruction execution times and failure rates are either highly, slightly or not correlated. This impacts the heterogeneity of the machine qualities. The number of instances is hence $3 \times (100 + 15) = 345$.

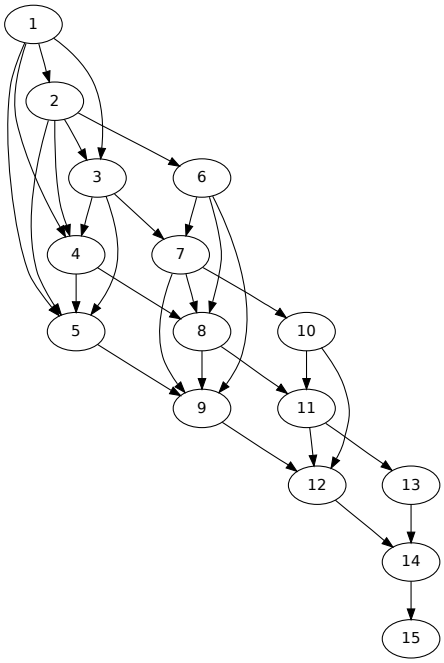
For independent tasks, the input cases are the following. We consider the same set of machines as above. We generate sets of tasks with cardinality between 10 and 100 (by increment of 1). For each cardinality of tasks, we build 4 different input classes. Each classes varies in function of the bound of the processing requirement. Indeed, we draw the processing requirement of the tasks uniformly between 1 and $\mathcal{B}$ where $\mathcal{B}$ is 100, 10^4, 10^6 or 10^9 depending on the class. Hence, we have $91 \times 4 = 364$ instances.

B. Parameters Study

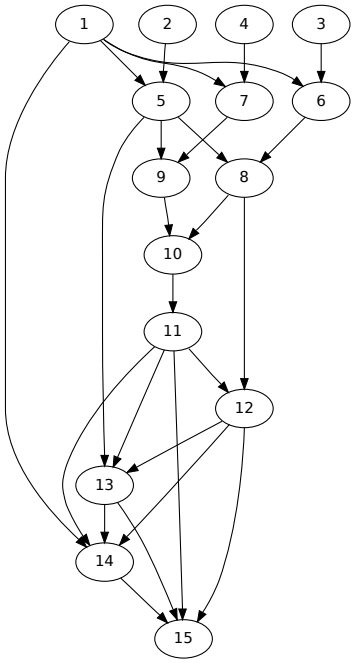
We study the effect of the parameters l (the number of elements generated from a partial solution) and k (the maximum size of the partial solution set). We use the set preference relation $\preccurlyeq_H$ to select solutions and execute the MO-GREEDY on all the combinations of values between 2 and 200. For this study, we select a subset of 12 instances for each problem. As relation $\preccurlyeq_H$ is used, it is reasonable to consider the hypervolume for measuring the quality of each generated Pareto set approximation (*i.e.*, the dominated area of an approximation with respect to a reference point). This reference point is selected by considering the first objective value of the best solution for the second objective and the second objective value of the best solution for the first objective.

For each solution, the corresponding hypervolume is normalized. This is required for comparing the hypervolume values obtained using distinct instances because this indicator is sensitive to the inputs such as the the task costs. Hence, we divide it by the hypervolume obtained through the execution of a comparable run (same set of parameters) but with the value of k leading to the best approximation.

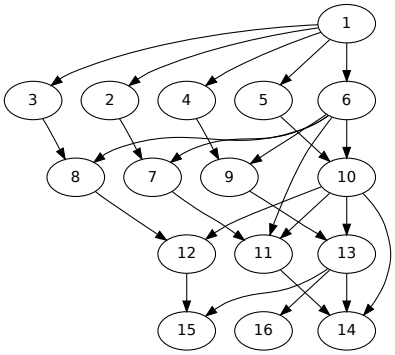
In Figures 5a and 5b, measures are depicted with boxplots (five-number summary: the extreme of the



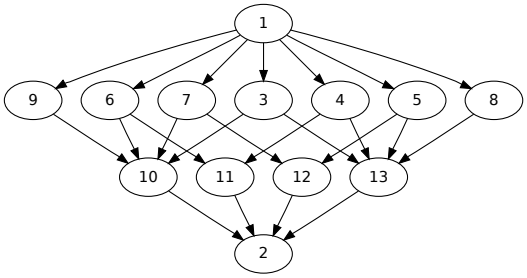
(a) Cholesky decomposition with 15 tasks.



(b) Gaussian elimination followed by a backward substitution with 15 tasks.



(c) Gaussian elimination with 16 tasks.



(d) Strassen matrix multiplication with 13 tasks (recursion on one level).

Figure 4: Examples of task graphs scheduled with the MO-GREEDY strategy.

lower whisker, the first quartile, the median, the third quartile and the extreme of the upper whisker). Each whisker extends to the most extreme data point that is no more than 1.5 times the interquartile range from the closest quartile. Values of any data points that lies beyond the extremes of the whiskers are represented by points (the outliers).

Figure 5a shows the influence of the value of k. We see that the median behavior increases with the value of k. This is an intuitive result as with an unbounded set of partial solutions, our method degenerates into an exhaustive search, which produces the Pareto front.

Figure 5b shows the influence of the value of l. The best median hypervolume is achieved with a subpopu-

lation of 10. It is a counter-intuitive result as we could have expected that the MO-GREEDY implementation would produce a better result as the value of l increases. This could be resulting from the imperfections of the shifted makespan metric, which provides only estimations on the quality of partial solutions.

C. Comparison with Other Approaches for the Scheduling Problem

For comparing the MO-GREEDY approach to other approaches, we have used: the HSA ranking [11] to order tasks, $k = 50$, $l = 10$ and the set preference relation $\preccurlyeq_H$. Other multi-objective heuristics are an aggregation-based one [11] (*BSA*), a heuristic where only a subset of the processors are used depending on the targeted compromise [8] (*Proc*), and one based on a geometrical compromise of the objectives such as it is done in [5] (*Geom*). These heuristics are thoroughly described in the appendix (Section VIII). They have been chosen because they have all been shown to be good heuristics.

In this section, data representation is done through histograms and empirical cumulative distribution functions (ECDF) that represent the statistical dispersion of the measures (see Figure 6). On the lower part of the figure, each line and column intersects into an ECDF and the corresponding histogram. These plots depict the repartition of the hypervolume ratios that are obtained by dividing the hypervolume obtained by the heuristic labelled on the line by the hypervolume of the one labelled on the column (labels are positioned on the diagonal). On the upper part, each numeric summary indicates: the proportion of ratios that are strictly above or below 1; the median ratios; and the proportion of ratios that are equal to 1 (if any). We recall that the hypervolume represents the dominated area, and hence is to be maximized. For example, in Figure 6, we see that GEOM achieves a better hypervolume than BSA does in 66.63% of the cases (the median ratio is 2.047).

On Figure 6, we see that geometrical search is better than aggregation and that using *Proc* leads to very good results. However, the best overall method is our MO-GREEDY implementation: it is better in 59.81% of the cases and on the median case.

The authors of [13] proposed a $(2 + \epsilon, 1)$ approximation algorithm of the Pareto front (makespan vs. failure probability) for the scheduling problem in case of independent tasks. The cardinality of the generated set is $O(1/\epsilon)$ and the computational complexity is proportional to $1/\log(1 + \frac{2}{\epsilon})$. We have implemented this algorithm and compared the results against the MO-GREEDY implementation with the set preference relation $\preccurlyeq_H$. In Figure 7a, we compare the $\epsilon = 0.001$ case for the approximation algorithm against the $k = 50$ $l = 5$ case for the MO-GREEDY implementation. The histogram and the ECDF of the hypervolume ratios between the MO-GREEDY approach and the approximation algorithm is plotted. When this ratio is greater than one, the MO-GREEDY outperforms the approximation algorithm for this indicator. The median case is 1.012 and more than 68.2% of the ratios are greater than 1. Therefore, with these settings, the MO-GREEDY approach outperforms the approximation algorithm. This means that in at least 2-third of the cases, the MO-GREEDY approach is able to deliver a front where there are points that have a makespan at most 2.001 larger than a Pareto optimal point and a reliability no worse than this point.

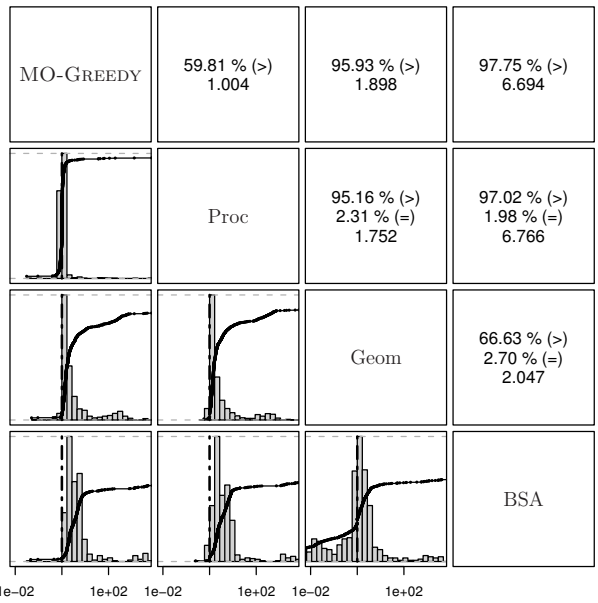
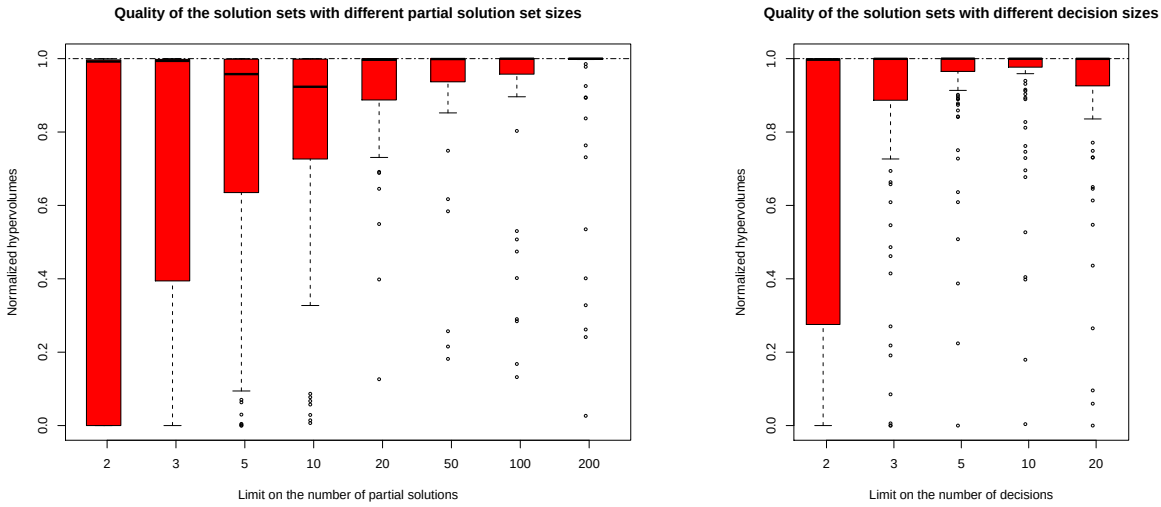


Figure 6: Comparison of the hypervolume ratios between four multi-objective heuristics.

In Figure 7b, we compare the $\epsilon = 0.01$ case for the approximation algorithm against the $k = 200$ $l = 20$ case for the MO-GREEDY implementation. In this case, we see that the result is similar to the previous case with a median case of 1.01 and more than 66.2% of the ratios are better than 1.

Last, in Figure 7c, we compare the $\epsilon = 0.0001$ case for the approximation algorithm against the $k = 1000$ $l = 20$ case for the MO-GREEDY implementation. In this case, we see that MO-GREEDY always outperforms the approximation algorithm.

The conclusion of these experiments is that for independent tasks, as for the task-graph case shown in Figure 5, the larger the number of partial solutions the better the front.



(a) Variation of the size of the set of partial solutions that are considered at each step.

(b) Variation of the number of decisions that are selected while building partial solutions.

Figure 5: Qualities of the MO-GREEDY outputs with various values for both size limits, k and l. The quality is based on the hypervolume normalized by the best median case (the higher, the better). Measures for $l > 20$ are not depicted because they are equivalent to the case $l = 20$ (as there are 20 processors).

VI. RELATED WORK

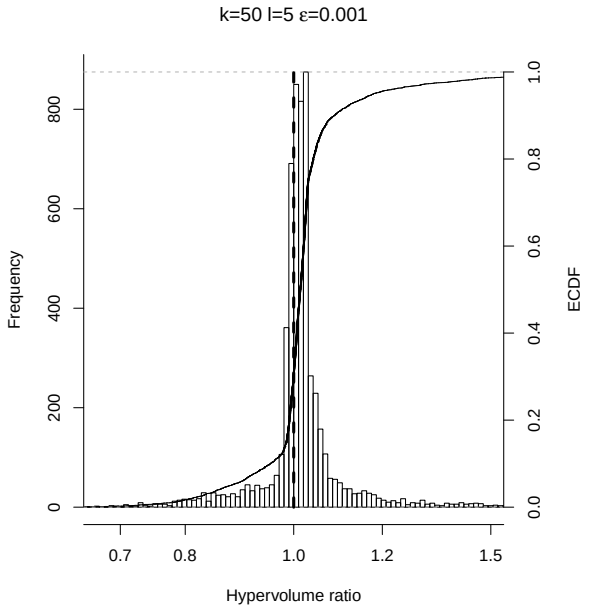
To compute a set of solutions in the multi-objective case, several approaches exist. Genetic Algorithms (GA) deal with the multi-objective case. Several frameworks offer a simplified way to implement multi-objective GA (*e.g.*, ParadisEO [4] and PISA [2]). However, as for single-objective GA, the computation time is unbounded and can become high if high quality solutions are required.

Aggregation is an intuitive technique that consists in finding a trade-off between objectives by computing a multi-parameter formula of these objectives (*e.g.*, $\theta c_1 + (1-\theta)c_2$ for the bi-objective case, where c_1 (resp., c_2) is the value of the first (resp., second) objective). By varying the compromise parameter θ between 0 and 1, one can expect to cover the whole front while applying the same algorithm. This method is fast and efficient when carefully done (*e.g.*, using normalization), but does not work well when the objective space near the Pareto front is concave.

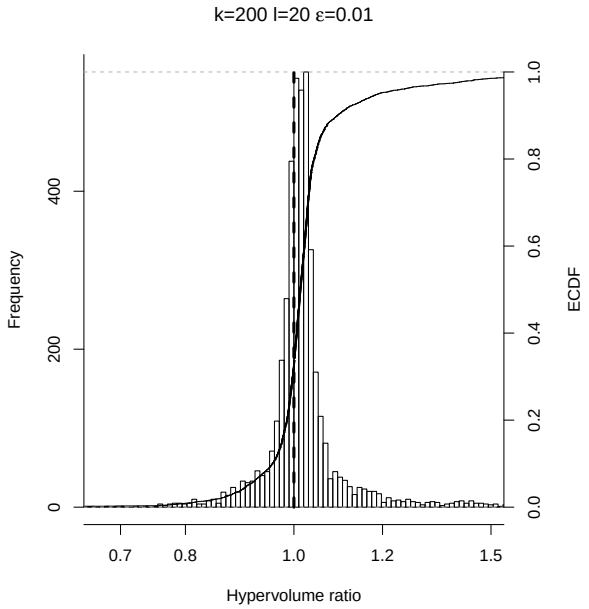
This work is also related to decision tree search (*e.g.*, branch-and-bound or A*) as greedy algorithm can be seen as walking in a tree. However, these techniques are known to be time-consuming and usually seek either for one solution only or for the entire Pareto front. As the number of Pareto-optimal solutions can be exponentially high, it lacks techniques that are able to prune the search and sample the solution space.

A multi-objective iterative greedy search for bi-objective flowshop problem was proposed in [10]. This method relies on building a front of non-dominated solution by iteratively modifying the current solution set (initialized by single-objective heuristics). The main difference with our approach is that it is not as general and is only suited to the case when input objects are not explicitly ranked: it is not applicable for a scheduling problem with precedence constraints, because in this case only tasks at the bottom of the graph could be moved from one processor to another.

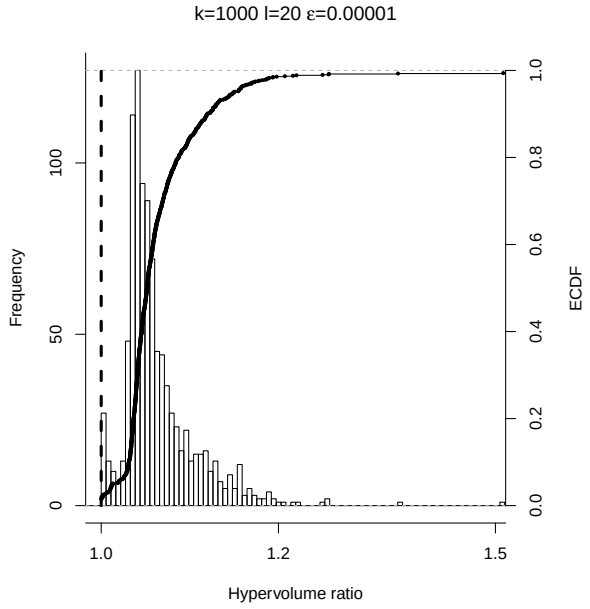
Beam-search was first use in speech recognition [15]. There exists several results that use beam-search in scheduling or for multi-criteria problem. In [17], beam-search is used for a bi-criteria train scheduling problem, however, in this case, the selection of the candidates node is done using an aggregation of the different criteria. In [16], beam-search is used for a multi-criteria scheduling problem in an assembly line. In this case the selection of the candidates node is based on a single criterion (only nodes with the best usage rates are kept). In [7], a bi-criteria scheduling problem for single machine is studied. In this case, the selection criteria is an aggregation of both criteria. Other studies [14], [9], [1] use also beam-search for solving multi-criteria scheduling problem. However, to the best of our knowledge, none of the existing works select the candidate nodes using an evaluation of the current Pareto front as we do here with the set preference relation $\preccurlyeq_H$.



(a) Small number of partial solution for the MO-GREEDY strategy and medium ϵ.



(b) Medium number of partial solution for the MO-GREEDY strategy and large ϵ.



(c) Large number of partial solutions for the MO-GREEDY strategy and small ϵ.

Figure 7: Comparison of the hypervolume ratios between the MO-GREEDY strategy and an approximation algorithm for the independent tasks case.

VII. CONCLUSION

Computing or approximating the Pareto front is one of the major challenge in multi-objective optimization. Moreover, finding a good solution rapidly is a property of many greedy algorithms. In this paper, we have proposed a generalization of the greedy approach, called MO-GREEDY, for the multi-objective case. This method is an extension of the beam-search technique.

A multi-objective problem for which there exists a greedy heuristic for the single-objective case can then, after some adaptation, be solved thanks to our method. Similarly to the greedy approach, adaptation to the targeted problem is required.

We have studied the quality of our strategy for a bi-criteria scheduling problem for heterogeneous computing and discussed the impact of different possible settings (limit to the size of the manipulated sets). Moreover, experimental results show that the proposed strategy produces solutions that are close to Pareto-optimal ones and outperforms aggregation or efficient specialized multi-objective heuristics in many cases.

Our future work is directed towards the implementation of problems with more objectives. However, existing methods for computing iteratively the hypervolume pose practical issues. We also want to investigate how we could introduce algorithmic guarantees for specific problems such as it is done for greedy algorithms.

REFERENCES

[1] Meral Azizoglu, Suna Kondakci, and Ömer Kirca. Bicriteria scheduling problem involving total tardiness and total earliness penalties. *International Journal of Production Economics*, 23(1-3):17 – 24, 1991.

[2] Stefan Bleuler, Marco Laumanns, Lothar Thiele, and Eckart Zitzler. PISA — a platform and programming language independent interface for search algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, Lecture Notes in Computer Science, pages 494 – 508, Berlin, 2003.

[3] Lucas Bradstreet, Lyndon While, and Luigi Barone. A Fast Incremental Hypervolume Algorithm. *IEEE Transactions on Evolutionary Computation*, 12(6):714–723, December 2008.

[4] Sébastien Cahon, Nordine Melab, and El-Ghazali Talbi. ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004.

[5] Louis-Claude Canon and Emmanuel Jeannot. Evaluation and optimization of the robustness of dag schedules in heterogeneous environments. *IEEE Transactions on Parallel and Distributed Systems*, To appear.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. MIT Press and McGraw-Hill, 2002.

[7] Prabuddha De, Jay B. Ghosh, and Charles E. Wells. Heuristic estimation of the efficient frontier for a bi-criteria scheduling problem. *Decision Sciences*, 23(3):596–609, 1992.

[8] Jack J. Dongarra, Emmanuel Jeannot, Erik Saule, and Zhiao Shi. Bi-objective Scheduling Algorithms for Optimizing Makespan and Reliability on Heterogeneous Systems. In *19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'07)*, San Diego, CA, USA, June 2007.

[9] Fatih Safa Erenay, Ihsan Sabuncuoglu, Aysegül Toptal, and Manoj Kumar Tiwari. New solution methods for single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs. *European Journal of Operational Research*, 201(1):89 – 98, 2010.

[10] Jose Framinan and Rainer Leisten. A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum*, 30(4):787–804, October 2008.

[11] Mourad Hakem and Franck Butelle. Reliability and scheduling on systems subject to failures. In *ICPP '07: Proceedings of the 2007 International Conference on Parallel Processing*, page 38, Washington, DC, USA, 2007. IEEE Computer Society.

[12] Salim Hariri Haluk Topcuoglu and Min-You Wu. Task scheduling algorithms for heterogeneous processors. *8th IEEE Heterogeneous Computing Workshop (HCW'99)*, pages 3–14, April 1999.

[13] Emmanuel Jeannot, Erik Saule, and Denis Trystram. Bi-Objective Approximation Scheme for Makespan and Reliability Optimization on Uniform Parallel Machines. In *The 14th International Euro-Par Conference on Parallel and Distributed Computing (Euro-Par 2008)*, Las Palmas de Gran Canaria, Spain, August 2008.

[14] Yow-yuh Leu, Philip Huang, and Roberta Russell. Using beam search techniques for sequencing mixed-model assembly lines. *Annals of Operations Research*, 70:379–397, 1997. 10.1023/A:1018938608304.

[15] Bruce T. Lowerre. *The Harpy Speech Recognition System.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1976.

[16] P.R. McMullen and Peter Tarasewich. A beam search heuristic method for mixed-model scheduling with setups. *International Journal of Production Economics*, 96(2):273 – 283, 2005.

[17] Xuesong Zhou and Ming Zhong. Bicriteria train scheduling for high-speed passenger railroad planning applications. *European Journal of Operational Research*, 167(3):752 – 771, 2005. Multicriteria Scheduling.

[18] Eckart Zitzler, Lothar Thiele, and Johannes Bader. On Set-Based Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 2010. To appear.

[19] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.

VIII. Appendix: Heuristics Solving the Scheduling Problem

A. Proc: Approximating the Pareto Front Using a Makespan-Centric Heuristic

It has been shown in [8] that scheduling tasks on the processors with the smallest $\lambda\tau$ helps in improving the reliability. The *Proc* heuristic uses this fact to provide an approximation of the Pareto front.

Algorithm 3: Proc: a heuristic for approximating the Pareto-front

Data: G the input DAG
Result: S an approximation of the Pareto-front
begin
  Sort the processors in non-decreasing $\lambda_j\tau_j$ order $S \leftarrow \emptyset$ **for** j *from* 1 *to* m **do**
    Let π_j be the schedule of G obtained by HEFT using the first j processors **if** π_j *is not dominated by any solution of* S **then**
      $S \leftarrow S \cup \{\pi_j\}$
  return S

The idea is to build a set of makespan/reliability trade-offs by scheduling the tasks on a subset of processors (sorted by non-decreasing $\lambda\tau$ product) using the HEFT heuristic. The smaller the number of used processors, the smaller the makespan and the better the reliability (and vice-a-versa).

B. BSA: Bi-objective Aggregation-based Heuristic

BSA [11] is a heuristic based on aggregation that uses an additive function to map tasks. Given a ranking of the tasks, the heuristic schedules task t_i to the processor p_j such that:

$$\sqrt{\theta\left(\frac{\text{end}(i,j)}{\max_{j'}\text{end}(i,j')}\right)^2 + (1-\theta)\left(\frac{o_i\tau_j\lambda_j}{\max_{j'} o_i\tau_{j'}\lambda_{j'}}\right)^2}$$

is minimized, where, $\text{end}(i,j)$ is the completion time of task i if it is scheduled as soon as possible on processor j and θ is the parameter given by the decision maker that determines the tradeoff between each objective ($\theta = 1$ leads to makespan centric heuristic). Each term represents one of the objective and is normalized since all objectives are expressed in different units and can have different orders of magnitude. The normalization is done relatively to an approximation of the worst allocation of the tasks.

Different values of θ are used (from 0 to 1 by 0.01 increment) to build the Pareto front.

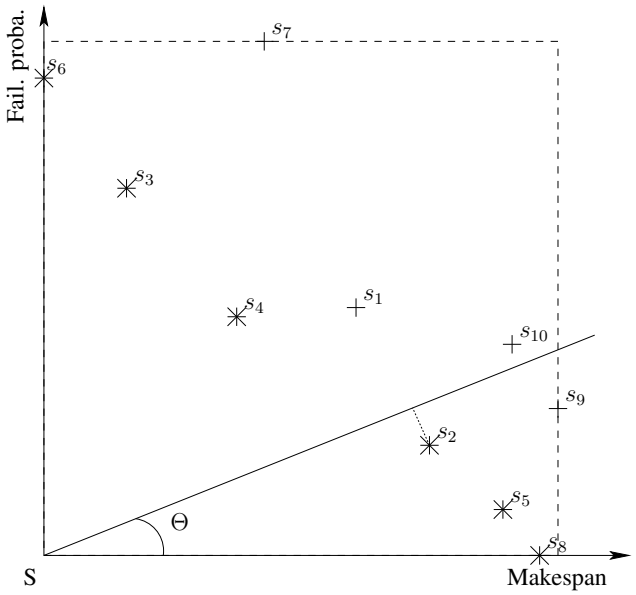


Figure 8: The geometric heuristic with 10 processors: stars are Parto-optimal solutions and crosses are dominated solutions, which are discarded. Hence, partial schedule s_2 is selected and the task is mapped on processor 2.

C. Geom: Bi-objective Geometric-based Heuristic

Concerning the geometric class of heuristics, the idea has been introduced in [5] and is described below. The decision maker provides an angle θ between $0°$ and $90°$ and a greedy scheduling algorithm. At each step, a partial schedule S is incremented and a new task is considered. The algorithm simulates its execution on all the m processors and hence, it generates m partial schedules, each one having its own reliability and makespan. Among these schedules, we discard the Pareto dominated ones. Then, these partial schedules and S – the one generated at the previous step – are plotted into a square of size 1, S being at the origin (see Figure 8). Then, a line determined by the origin and an angle θ with the x-axis is drawn. The closest partial schedule to this line is retained (s_2 in the figure) and we proceed the next step.

IX. Biographies

Louis-Claude Canon Louis-Claude Canon holds a postdoctoral position at the Ensimag. He received his Master degree in computer science in 2007 from both the ESEO (Ecole Supérieure d'Electronique de l'Ouest) and the University of Angers. He has worked for two years on his Ph.D. on uncertainty management in parallel systems at the Loria (Laboratoire Lorrain de Recherche en Informatique et ses Applications) before joining the LaBRI (Laboratoire Bordelais de Recherche

en Informatique) where he finalized it. He received its Ph.D. degree in computer science from the University of Nancy in 2010. He is now teaching and researching at the University of Grenoble. His main research interests include scheduling, multi-objective combinatorial optimization, and stochastic optimization.

Emmanuel Jeannot Emmanuel Jeannot is a research scientist at INRIA (Institut National de Recherche en Informatique et en Automatique) and he is doing his research at INRIA Bordeaux Sud-Ouest and at the LaBRI laboratory since Sept. 2009. Before that he holds the same position at INRIA Nancy Grand-Est. From Jan. 2006 to Jul. 2006, he was a visiting researcher at the University of Tennessee, ICL laboratory. From Sept. 1999 to Sept. 2005, he was assistant professor at the Université Henry Poincaré, Nancy 1. During the period of 2000 to 2009, he did his research at the LORIA laboratory. He got his PhD and Master degree in computer science (resp., in 1996 and 1999) both from Ecole Normale Supérieur de Lyon, at the LIP laboratory. After his PhD, he spent one year as a postdoc at the LaBRI laboratory in Bordeaux. His main research interests are scheduling for heterogeneous environments and grids, data redistribution, algorithms and models for parallel machines, grid computing software, adaptive online compression and programming models.