Decoder-based Evolutionary Algorithm for bi-objective Just-in-Time Single-Machine Job-shop

Sophie Jacquin, Emilie Allart,Fanny Dufossé, Laetitia Jourdan
Inria Lille - Nord Europe, DOLPHIN, CRIStAL, University of Lille 1. Villeneuve d'Ascq, France.
sophie.jacquin@inria.fr, laetitia.jourdan@univ-lille1.fr, fanny.dufosse@inria.fr

***Abstract*—The bi-objective just-in-time single-machine job-shop scheduling problem (JIT-JSP) aims at simultaneously minimizing earliness and tardiness. In this paper, a multi-objective decoder-based evolutionary algorithm is proposed. The decoding strategy divides the search into two steps. In the first step, the search of the permutation order of the jobs is realized thanks to a multi-objective evolutionary algorithm. For a fixed permutation, the decoder algorithm optimizes the multi-objective timing sub-problem in the second step. Thus each permutation order induces a Pareto set of solutions. Two different decoding strategies to fix the idle times are proposed, one approximate and one exact. A comparison study with a classical multi-objective evolutionary algorithm underlines the performance of the proposed decoding strategy and the interest of the approximate decoder.**

I. Introduction

Job-shop scheduling problems, where multiple jobs have to be executed on one or several machines, are widely studied in academic area [1] as they have a lot of industrial applications. In particular, many manufacturing systems adopt the Just-in-Time production philosophy [2]. Their objective is to develop scheduling policies that minimize due date deviations of each job. Indeed, finishing a job before or after its due date involves direct and indirect costs. For example, inventory carrying costs, such as storage and insurance costs arise when jobs are finished before their due dates. Likewise, a job finishing late can incur shortage costs going from back order or transportation expediting costs to the loss of an important customer.

For this reason the Just-In-Time single-machine Job-Shop scheduling Problem (JIT-JSP) has attracted considerable research attention of the scheduling community. The problem is mostly modeled as a single objective problem that minimizes the weighted sum of total earliness and total tardiness of jobs. Both exact and heuristic methods have been proposed to solve this problem. The branch and bound method is especially prominent among exact approaches. In [3], Shaler compares several branch and bound procedures. Sourd *et al.* combine branch and bound with Lagrangian relaxation [2]. The authors of [4] propose a hybridization of column generation, Lagrangian relaxation, and dynamic programming. Due to the complexity of this problem, metaheuristic algorithms have also been applied, in particular local searches that give excellent results [5], [6], [7].

However, the bi-objective version of the JIT-JSP has been little studied, even though considering objectives of minimizing earliness and tardiness independently can be advantageous. It could be difficult for the decision maker to design penalty values for both earliness and tardiness in the same scale. Indeed, the monetary cost of a deviation from a due date is not easy to model as a simple function of job earliness or tardiness. Especially the tardiness penalty is often evaluated according to the risk of dissatisfaction of the customer, while the earliness penalty models storage costs. Therefore it is more convenient for the decision maker to evaluate the penalties of earliness and tardiness independently. Moreover the two objectives are clearly conflicting, which makes it interesting to study the trade-offs among the two.

To the best of our knowledge, the study of Rahimi-Vahed *et al.* [8] is the only one that considers the earliness and tardiness as two separate objectives to be minimized simultaneously. Authors assume that allowing machine idle time is not suitable. However, the earliness performance measure is non-regular and, for problems with unequal due dates, the insertion of idle times may reduce the costs of a schedule [9]. Therefore, decisions about completion times of jobs may be as important as decisions about the permutation of jobs.

In this paper, the bi-objective JIT-JSP with allowed idle times is studied. Three algorithms are designed and compared to solve the problem. The first one is based on the reference algorithm: Non-dominated Sorting Genetic Algorithm II (NSGA-II) [10] to fix the permutation order of the jobs. Genetic algorithm is the most popular type of evolutionary algorithm and mimics the metaphor of natural biological evolution. It operates on solutions called individuals that are defined by a genotype. The genotype is expressed by a phenotype that reflects the quality of the solution for a considered problem. The two other proposed algorithms are decoder-based hybridizations of NSGA-II with a timing procedure. Hence we consider two timing procedures. The proposal of a decoder-based hybridizations to solve the bi-objective JIT-JSP is the main contribution of this paper. A comparison analysis is performed to compare the three algorithms. This analysis allows us to evaluate the interest of the hybridization and the impact of the choice of the timing procedure.

This paper is organized as follows. First the bi-objective JIT-JSP is presented in details, then the three solving methods in consideration are presented. The experimental protocol is described in section IV and the results are shown and discussed section V. Finally, a conclusion about the potential of the proposed methods and perspectives is given.

II. The Just-In-Time Single-Machine Job-Shop Scheduling Problem

The Just-In-Time Job-Shop Scheduling Problem has been proposed in [2] and its objective is to schedule a set of N jobs $J = \{J_1, J_2, ..., J_N\}$ on a single machine such that there earliness and tardiness are minimized. We assume that the machine is continuously available and can not process more than one job at a time (see formula 4). Each job j should be scheduled before its release date r_j (see formula 3), takes a processing time p_j to be executed, and must be finished at its due date d_j. The earliness of a schedule is the sum of the job's earliness weighted by parameter α_j (see formula 1) and the tardiness is the sum of the job's tardiness weighted by parameter β_j (see formula 2).

Formally, the JIT-JSP aims at fixing variable C_j, the completion times of jobs ($j \in \{1...N\}$), such that earliness (1) and tardiness (2) objectives are minimized and release (3) and non-overlap (4) constraints are met.

For a given scheduling C, the value of earliness $E(C)$ and tardiness $T(C)$ can be expressed as:

$$E(C) = \sum_{j=1}^{N} \alpha_j \times max(0, d_j - C_j) \tag{1}$$

$$T(C) = \sum_{j=1}^{N} \beta_j \times max(0, C_j - d_j) \tag{2}$$

A scheduling C must respect the following constraints:

$$C_j - p_j \geq r_j \quad \forall j \tag{3}$$

$$[C_j - p_j, C_j] \cap [C_i - p_i, C_i] = \emptyset \quad \forall i, j, i \neq j \tag{4}$$

The set of scheduling solutions is denoted Ω and is represented in the objective space by a pair $(E(C), T(C))$ for a solution C in Ω.

The JIT-JSP has been proved NP-hard even when the objectives are aggregate [11].

The feasible solutions of the two dimensional *objective space* are compared using the *Pareto dominance* $\succ$. In this minimization context, a solution $x \in \Omega$ is said to dominate a solution $y \in \Omega$, denoted by $x \succ y$, if they satisfy relation (5).

$$\forall i \in \{1,2\},\ f_i(x) \leq f_i(y) \bigwedge \exists i \in \{1,2\},\ f_i(x) < f_i(y) \tag{5}$$

where f_1 is E and f_2 is T.

Table I summarizes parameters and variables of JIT-JSP problem.

III. Solving methods

Three evolutionary algorithms based on the Non-dominated Sorting Genetic Algorithm II are implemented and compared. The purpose of this analysis is first to study the impact of an indirect representation (permutation without idle times). Then, in case where an indirect representation is chosen, two decoding strategies are compared.

Notation	Definition
J_i	Job number i
r_i	Release time of job J_i
p_i	Processing time of job J_i
d_i	Deadline of job J_i
α_i	Earliness penalty of job J_i
β_i	Tardiness penalty of job J_i
C_i	Completion time of job J_i in schedule C
$E(C)$	Earliness of schedule C
$T(C)$	Tardiness of schedule C

TABLE I
Table of notations for JIT-JSP.

NSGA-II is presented in details in section III-A. Then a basic application of NSGA-II to the bi-objective JIT-JSP is described section III-B. In this method, a solution of the genetic algorithm, called genotype, gives a complete representation of a JIT-JSP solution, i.e. both jobs permutation order and idle times are encoded in the solution. In section III-C, a decoder-based hybridization between NSGA-II and a timing procedure, called D-NSGA-II, is proposed. D-NSGA-II uses an indirect representation where only jobs permutation order is described by a genotype and idle times are fixed by a decoder based on a timing procedure. Two timing procedures are proposed: one heuristic and one optimal.

A. NSGA-II: Non-dominated Sorting Genetic Algorithm II

NSGA-II was proposed by Deb *et al.* [10]. This evolutionary approach builds a population of competing individuals (genotype), ranks and sorts each individual according to its non-domination level, applies evolutionary operators to create a new offspring pool, and then combines the parents and offspring before partitioning the new combined pool into fronts. For each ranking level, a crowding distance is estimated by calculating the sum of distances between the two neighboring solutions for both objectives. Once the non-domination rank and the crowding distance is calculated, the surviving individuals are determined using the crowded-comparison operator ($\prec_n$). The crowded-comparison operator guides the selection process at the various stages of the algorithm toward an uniformly spread-out Pareto optimal front. Assuming that every individual in the population has two attributes: 1) non-domination rank (i_{rank}) and 2) crowding distance ($i_{distance}$), the partial order $\prec_n$ is defined as:

$$\begin{aligned} i \prec_n j: \quad & \text{if } (i_{rank} < j_{rank}) \text{or} \\ & ((i_{rank} = j_{rank}) \text{ and } (i_{distance} \geq j_{distance})) \end{aligned} \tag{6}$$

That is, between two solutions with differing non-domination ranks, we prefer the solution with the lower (better) rank. Otherwise, if both solutions belong to the same front, then the solution that is located in a less crowded region is preferred. Algorithm 1 presents the outline of the NSGA-II, which (in the last decade) has been the most popular Multi-Objective Evolutionary Algorithm (MOEA), and is frequently adopted to compare the performance of newly introduced MOEAs.

Algorithm 1: General Framework of NSGA-II

Input:
N: the population size;
stopping_condition: A stopping criterion;
Output:
P_t: the final approximation of the Pareto front;

```
1  begin
2      t = 0;
3      Generate a random population P_t of size N;
4      Evaluate the population P_t;
5      while stopping_condition is not satisfied do
6          Generate the offspring population Q_t;
7          Evaluate the offspring population Q_t;
8          R_t = P_t ∪ Q_t;
9          Rank R_t by using non-dominated sorting to define F; //
           F = (F_1, F_2, ...), all non-dominated fronts of R_t
10         P_{t+1} = ∅ and i = 1;
11         while (|P_{t+1}| + |F_i| ≤ N) do
12             Assign crowding distance to each front F_i;
13             P_{t+1} = P_{t+1} ∪ F_i;
14             i = i + 1;
15         Sort F_i by using the crowded-comparison operator;
16         P_{t+1} = P_{t+1} ∪ F_i[1 : (N − |P_{t+1}|)];
17         t = t + 1;
18     return P_t;
```

B. NSGA-II applied to the JIT-JSP

This section details the application of NSGA-II to the bi-objective JIT-JSP. The choices of genotypic representation, initialization process and evolutionary operators are justified.

1) Representation: A solution is as a sequence of blocks separated by idle times. A block B_i is an ordered sequence of jobs that are executed consecutively without any idle time as illustrated in Figure 1. Only feasible solutions can be represented. In the following, the j^{th} jobs of the i^{th} block, B_i, is denoted J^i_j and the idle time above B_i is denoted Id_i. $|B_i|$ denotes the number of jobs in block B_i.

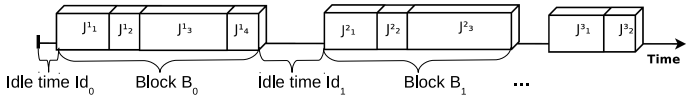

Fig. 1. Genotypic representation of a solution for the basic NSGA-II.

2) Initialization: The process of building an individual of the initial population is the following. First, π, the permutation order of jobs, is randomly generated. Then the idle times are fixed by solving the aggregate timing sub-problem $TP_\alpha(\pi)$ for α randomly selected in $[0, 1]$:

$$TP_\alpha(\pi)\begin{cases} \min\limits_{C}(\alpha E(C) + (1-\alpha)T(C)) & \\ s.t: & \\ C_{\pi_i - 1} \leq C_{\pi_i} + p_{\pi_i} & \forall i \in \{2...N\} \\ C_j \geq r_j + p_j & \forall j \in J \end{cases}$$

$TP_\alpha(\pi)$ is solved optimally in $O(N\ln(N))$ by using Algorithm 2. It corresponds to the algorithm of Bauman *et al.* [12] but several modifications have been done. The first one modifies a portion of the original algorithm [12] and is underlined in dark gray in Algorithm 2. It aims at forcing release constraints to be met. The criteria for applying the left shift procedure has also been changed (line 22 in light gray box in Algorithm 2). Thanks to that, the left shift procedure is only applied if it *strictly* improves the quality of the solution instead of being applied if it does not decrease its quality. This guarantee that solutions obtained for $TP_0(\pi)$ are Pareto optimal.

3) Mutation operators: Mutation operators are based on the neighborhood definitions proposed by Sourd for a mono-objective JIT-JSP [5]. We choose this study as a reference because it is, to the best of our knowledge, the only work that includes idle times in the representation. Three mutation operators are used, the two first ones modify the permutation order of jobs inside one block and the last one modifies the idle times with the possibility to create a new block or to merge two existing blocks.

- **Block swap mutation:** It consists in randomly selecting a bloc, B_i, and swapping two jobs randomly chosen inside this block.
- **Block extract and reinsert mutation:** It extracts one job randomly selected in a block B_i and reinserts it in an other place in the same block.
- **Subblock move mutation:** It aims at modifying or inserting idle times. A job j is randomly selected. Suppose that j is the k^{th} jobs of block i (i.e. $j = J^i_k$). So one of the following actions is randomly chosen:
  - Shift left process: Randomly shift left the subblock including jobs from J^i_0 to J^i_k. The maximum shift is Id_i such that the starting times of the other blocks remain unchanged. This process is illustrated by Figure 2.
  - Shift right process: Randomly shift right the subblock including jobs from J^i_k to $J^i_{|B_i|}$. The maximum shift is Id_{i+1} such that the starting times of the other blocks remain unchanged. In case where B_i is the last block (i.e. $B_i = B_{n_b}$), the maximum shift is fixed to $(\max_i(d_i) - C_{J^{n_b}_0})$.

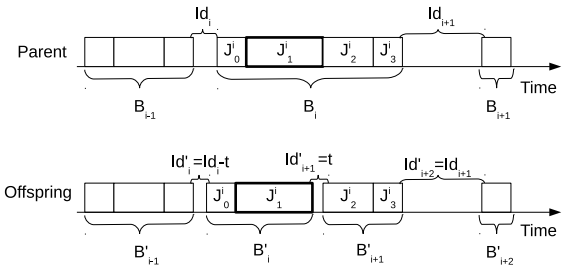

Fig. 2. Subblock move mutation : Job J^i_1, shift left process and $t \in [0, Id_i]$ are randomly selected. Then jobs J^i_0 and J^i_1 are shift left of t.

4) Crossover operator: The crossover operator is an adaptation of the 2-points crossover. Indeed, the analysis study

Algorithm 2: Solving method for the aggregate timing problem.

Input:
π: jobs permutation order;
Output:
C: completion times of jobs;

begin
  $C_0 := 0; l := 0; H_0 := 0; \gamma_0 := 0; P_0 := 0;$
  for $k := 1$ *to* N **do**
    $x := \max(r_k, C_{k-1}) + p_k - d_k;$
    $P_k = \begin{cases} r_k + p_k & \text{if } r_k > P_{k-1} \\ P_{k-1} + p_k & \text{else.} \end{cases}$;
    $compMax := min(0, rk - C_{k-1});$
    if $H_l < compMax$ **then**
      $diff := H_l - compMax;$
      for $l := k$ *to* 0 **do**
        $H_l := \max(H_l - diff, 0);$
    if $x \leq 0$ **then**
      if $x < 0$ **then**
        $l := l + 1;$
        $H_l := H_{l-1} + x;$
        $\gamma_l := 0;$
      $\gamma_l := gamma_l + \alpha_k$;
      $C_k := d_k;$
    else
      $H_{new} := H_l + x;$
      if $H_{new} < 0$ **then**
        Insert H_{new} in H ;
        if *there is* H_j *such that* $H_{new} = H_j$ **then**
          Do not create a point but $\gamma_j := \gamma_j + \gamma_{new}$;
        else
          $\gamma_{new} = \alpha_k + \beta_k;$
          $l := l + 1;$
        $\gamma_l := \gamma_l - \beta_k;$
        $i := l;$
        while $\gamma_i < 0$ *and* $i > 0$ **do**
          $\gamma_{i-1} := \gamma_{i-1} + \gamma_i;$
          $C_k := P_k - H_{i-1};$
          $i := i - 1;$
          $l := l - 1;$
    $avail := C_N;$
    for $k := N$ *to* 0 **do**
      if $C_k \geq avail$ **then**
        $C_k := avail;$
      else
        $avail = C_k;$
      $avail := avail - p_{k-1};$
  return $C;$

presented in [13] demonstrates its efficiency in comparison to other classical operators CX (Cycle Crossover) and PMX (Partially Mapped Crossover). However, this 2-points crossover has been designed for Jobs-Shop Scheduling problems in which idle times are not allowed. To handle them, it is considered that each idle time is attached with the job that it precedes and then transmitted with it. In details, the 2-points crossover randomly selects two points. All the jobs among these two points in the first parent are inherited by the offspring as well as the corresponding idle times. Then the missing jobs are completed in their order of apparition in the second parent. Figure 3 illustrates this operator.

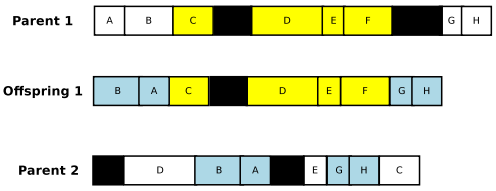

Fig. 3. 2-points crossover: The two points selected in the first parent are C and F. Then all points between C and F are inherited by offspring 1 from parent 1 with their idle times (in black). Then the missing jobs are completed in their order of apparition in the second parent. The second offspring is obtained by reversing the roles of the two parents.

C. Multi-objective decoder-based hybridization to the JIT-JSP

To solve the aggregate version of the JIT-JSP, most of metaheuristics from literature use a partial representation giving only the permutation order of the jobs while the idle times are fixed by a decoder [14], [15], [16]. This strategy gives very good results as it considerably reduces the search space of the metaheuristic. In this section, a similar strategy for the bi-objective JIT-JSP is studied. First we introduce the bi-objective Timing Problem, $TP(\pi)$, and its properties. Then a strategy to solve the bi-objective JIT-JSP with a decoder-based hybrid metaheuristic is detailed.

1) Timing problem: For a given permutation π, the timing problem $TP(\pi)$ is to fix the idle times in order to simultaneously minimize the earliness and tardiness objective functions. Formally, $TP(\pi)$ is formulated as follows:

$$TP(\pi) \begin{cases} & \min\limits_{C}(E(C), T(C)) & \\ s.t: & & \\ C_{\pi_i - 1} \leq & C_{\pi_i} + p_{\pi_i} & \forall i \in \{2...N\} \\ C_j \geq & r_j + p_j & \forall j \in J \end{cases}$$

As $TP(\pi)$ is a continuous optimization problem whose feasible solution set is convex as well as objective functions, it can be proved that its Pareto front is connected and convex [17]. This implies that all Pareto optimal solutions are supported and can be reached by the weighted sum method (i.e. C^* is a Pareto optimal solution if and only if it exists $\alpha^* \in [0, 1]$ such that C^* is a solution of $TP_{\alpha^*}(\pi)$). Thus, as for all $\alpha \in [0, 1]$, $TP_\alpha(\pi)$ can be solved in $O(N \ln(N))$, a possible method to approximate $TP(\pi)$'s Pareto front is to solve $TP_\alpha(\pi)$ with Algorithm 2 for all $\alpha \in \{0, \frac{1}{n_\alpha - 1}, \frac{2}{n_\alpha - 1}, ..., 1\}$. With this method, n_α, the number of points in the approximation, is a parameter to determine beforehand.

Moreover, as the objective functions are piecewise linear, it can be shown that $TP(\pi)$'s Pareto front is the union of finitely many semi-closed polyhedra [18]. So, the Pareto front of $TP(\pi)$ is represented in the objective space by a piecewise linear curve

as represented in Figure 4. We call extreme point of a problem $TP(\pi)$ an angle of the Pareto front.

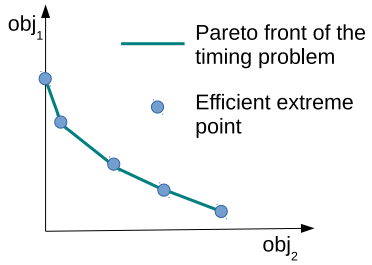

Fig. 4. Shape of the timing problem Pareto front in the objective space.

If $TP_\alpha(\pi)$ has many solutions, Algorithm 2 always gives the optimal solution with the smallest earliness. Then, only extreme points of the Pareto set can be found by the weighted sum method. Moreover, it is possible that, for a chosen n_α, some extreme points are not reached with any value $\alpha \in \{0, \frac{1}{n_\alpha}, \frac{2}{n_\alpha}, ..., 1\}$ or that several values of α give the same extreme point.

A method that insures to obtain the set of all the efficient extreme points is the Aneja-Nair method [19]. This method first computes $C_0(\pi)$ and $C_1(\pi)$, respectively the solutions of $TP_0(\pi)$ and $TP_1(\pi)$. Let recursively $(\alpha_i)_i$ be the ordering list of weights α_i that were already used by the algorithm to find efficient extreme points. At each iteration, a new weight, β_j, is constructed as the average of two weights α_j and α_{j+1} of $(\alpha_i)_i$. Then, if a new extreme point is provide $C_\beta(\pi)$, β is added to $(\alpha_i)_i$. The algorithm stops when there is no more j such that solving $TP_\beta(\pi)$ with $\beta = \frac{\alpha_j+\alpha_{j+1}}{2}$ gives a new efficient extreme point.

2) Representation and decoding: A solution is partially represented by a permutation order π while the idle times are fixed by a decoder that solves $TP(\pi)$. However, as $TP(\pi)$ is continuous and bi-objective, it has an infinite number of Pareto optimal solutions. So, there is no unambiguous way to create a single complete solution from a partial one. Figure 5 illustrates this situation.

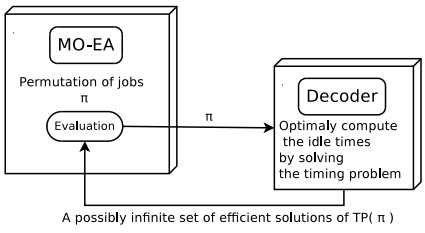

Fig. 5. Decoder-based hybridization for the bi-objective JIT-JSP.

A similar situation was previously studied for a multi-objective unit commitment problem [20]. In their work Jacquin et al. propose and compare three decoding strategies in case where the decoder is based on a multi-objective sub-problem. The two first strategies assign to each partial (genotypic) solution a single complete (phenotypic) solution. This is done by aggregating the objectives of the decoding problem. The third strategy assigns to each genotypic solution a set of solutions that approximates the Pareto front of the sub-problem. The statistical study performed in [20] indicates that the last strategy is significantly more efficient than the two others approaches. Consequently it is the one we choose to apply to JIT-JSP with a decoder-based hybridization. Then each permutation order (genotypic solution) is decoded by a set of phenotypic solutions.

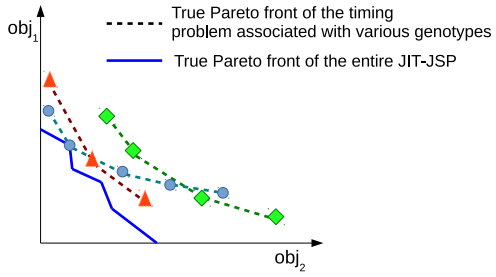

Fig. 6. Representation of three genotypic solutions in the objective space.

In Figure 6, three genotypic solutions are represented in the objective space. One is decoded by the set of square points, the other one by the set of triangular points and the last one by the set of round points. As a single genotype is represented by many points in the objective space, the fitness assignment and diversity assignment methods of NSGA-II have to be adapted. This adaptation is detailed, in the next section.

Two decoding strategies to approximate the Pareto front of $TP(\pi)$ are studied. The first one uses a weighted sum method, by solving $TP_\alpha(\pi)$ for all $\alpha \in \{\frac{0}{n_\alpha}, \frac{1}{n_\alpha}, ..., \frac{n_\alpha}{n_\alpha}\}$, with n_α a parameter of the algorithm.

The second one is to use the Aneja-Nair method [19]. This method gives the set of all the efficient extreme points which allows to obtain the exact Pareto front. But it can be more expensive in term of computation time than a simple use of weighted sum method.

In the following, the set of complete (or phenotypic) solutions obtained from a partial (or genotypic) solution π is denoted $D(\pi)$.

3) Adaptation on NSGA-II (D-NSGA-II): As a genotypic solution is decoded by a set of phenotypic solutions, the evaluation process of NSGA-II has to be adapted. More precisely, the convergence value (non-domination rank) and diversity value (crowding distance) used to evaluate a solution can not be applied any more. Then we use D-NSGA-II (Decoded based NSGA-II) that is proposed in [20].

In D-NSGA-II the evaluation process of NSGA-II is adapted to take into account that a single genotypic solution π is decoded by a set of phenotypic solutions $D(\pi)$. The convergence assigned to a genotypic (or partial) solution π is the best non-domination rank among the one of the phenotypic solutions $C \in D(\pi)$. Also, the diversity value assigned to a genotypic solution π, is the biggest crowding distance between

a phenotypic solutions $C \in D(\pi)$ and the phenotypic solutions $C' \notin D(\pi)$.

4) Evolutionary operators: To maintain similarity with the evolutionary operators (see section III-B), swap mutation and extract and reinsert mutation are re-used, except that, as there are no idle times in the representation, operators are applied as if there were a single block. Likewise 2-points crossover is re-used.

IV. Experimental Design

A statistical analysis is carried out to compare the performances of NSGA-II, D-NSGA-II with decoder based on the weighted sum (D-NSGA-II(W.S.)) and D-NSGA-II with decoder based on the Aneja-Nair method (D-NSGA-II(A.N.)). In this section the experimental design is presented.

A. Instances

The instances used for our experimental analysis are the ones proposed by Kerem et al. [21]. They are currently available online at http://www-poleia.lip6.fr/~safia/et/. Instances of 20, 40, 60, 80 and 100 jobs have been studied. In details, for a size of N jobs, 5 instances are studied respectively named bky$N$_60, bky$N$_120, bky$N$_180, bky$N$_240 and bky$N$_300.

B. Performance assessment

The different methods of performance assessment to compare multi-objective algorithms are explained in details in [22]. In our case the ε-indicator and the hypervolume difference indicator are selected for their complementarity. Let Z^{all} be the set of objective vectors from all the Pareto set approximations we obtained during all our experiments. Then, a reference set R contains the non-dominated points of Z^{all}.

a) ε-indicator $I^1_{\varepsilon+}$: The unary version of this indicator is computed using the binary version given by (7) and the reference set R, with $I^1_{\varepsilon+}(A) = I_{\varepsilon+}(A, R)$.

$$I_{\varepsilon+}(A, B) = \inf_{\varepsilon \in \mathbb{R}} \{\forall z^1 \in B, \exists z^2 \in A, \forall i \in 1 \dots n, z^1_i \leq \varepsilon + z^2_i\} \quad (7)$$

b) Hypervolume difference indicator I^-_H: The hypervolume indicator I_H is computed by the measure of the hypervolume between a set of solutions and the point $z = (z_1, \dots, z_n)$ where z_k is the upper bound of the k^{th} objective regarding all the solutions of Z^{all}. The hypervolume difference indicator I^-_H is then computed with $I^-_H(A) = I_H(R) - I_H(A)$.

C. Parameters setting

Proper settings of an evolutionary algorithm parameters are required to achieve the best performance. For this reason, a sensitivity analysis was carried out for NSGA-II and D-NSGA-II(W.S.) to determine the effect of the crossover rate, the mutations rates and the population size. In the case of D-NSGA-II(W.S.) the parameter n_α (number of points to approximate the Pareto front of the timing problem) has also been set. This analysis is done thanks to Irace [23] for each algorithm and each instance size. Irace is a package for R (a statistical software) that implements the iterated racing procedure. This procedure is an extension of the Iterated F-race procedure. Its main purpose is to automatically configure optimization algorithms by finding the most appropriate settings given a set of instances of an optimization problem. The parameters used for D-NSGA-II(A.N.) are the same than the ones used for D-NSGA-II(W.S.). In each case the stopping criterion used is a maximum execution time of N seconds.

D. Experimental setting

All the implementations are realized under the ParadisEO 2.0 [24] software framework. The programs have been implemented in C++, using the gcc 4.8.2 compiler and have been run on a desktop PC using Intel Pentium(R) Dual CPU T3200 @ 2.00GHz x 2 processors, under the Ubuntu 14.04 operating system.

For each instance 20 runs are performed with 20 different predefined populations. Then, as the samples are paired, the statistical analysis is based on Friedman statistical test with a $p - value < 0.01$ as criteria of rejection of the null hypothesis. The statistical tests are achieved using PISA [25] platform and its performance assessment module.

V. Results and discussion

Table II summarizes the results of the statistical tests obtained for the hypervolume difference indicator and Table III summarizes the results obtained for the ϵ-indicator.

In the tables the symbol "$\succ$" means that the method named in the row is statistically significantly better than the method named in the column for the chosen indicator. On the contrary, the symbol "$\prec$" means that the method named in the row is statistically less efficient than the method named in the column for the chosen indicator. Finally the symbol "$\simeq$" indicates an equivalence between the two methods.

It must first be noticed that for all size of instances D-NSGA-II is significantly better than the classical NSGA-II. This confirms the benefice of using a decoder-based hybridization.

However, the results of statistical comparisons between the two versions of D-NSGA-II varies from an instance to another. Indeed for small instances (20 and 40 jobs) the use of a decoder based on Aneja-Nair method gives better results than the use of a decoder based on the weighted sum method. But for bigger instances (60, 80 or 100 jobs) the decoder based on a simple weighted sum method gives results significantly better than the decoder based on Aneja-Nair method. This can be due to the fact that the approximation of the Pareto fronts of the timing sub-problems obtained with Aneja-Nair method are more precise and contain more points than the ones obtained with a simple weighted sum method, then the evaluation process in D-NSGA-II(A.N.) is more costly in term of computation time than the evaluation process of D-NSGA-II(W.S.). Actually, n_α the number of points compute by the weighted sum has been optimally determine by the *Irace* tuning process and, as show in Table IV, for the stopping criteria chosen (N seconds), this number is relatively low compare to the average number

			D-NSGA-II		NSGA-II
			W.S.	A.N.	
20 Jobs	I_{HD}	mean	0.5182	0.0461	0.6962
		std	0.0708	0.0014	0.0657
	D-NSGA-II	W.S.	X	$\prec$	$\succ$
		A.N.	$\succ$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X
40 Jobs	I_{HD}	mean	0.1438	0.35489	0.4848
		std	0.01066	0.0315	0.0387
	D-NSGA-II	W.S.	X	$\prec$	$\succ$
		A.N.	$\succ$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X
60 Jobs	I_{HD}	mean	0.096	0.1732	0.4203
		std	0.0079	0.005	0.0469
	D-NSGA-II	W.S.	X	$\succ$	$\succ$
		A.N.	$\prec$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X
80 Jobs	I_{HD}	mean	0.0792	0.2149	0.4063
		std	0.0048	0.0025	0.0651
	D-NSGA-II	W.S.	X	$\succ$	$\succ$
		A.N.	$\prec$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X
100 Jobs	I_{HD}	mean	0.14	0.2604	0.4805
		std	0.0065	0.0024	0.0361
	D-NSGA-II	W.S.	X	$\succ$	$\succ$
		A.N.	$\prec$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X

TABLE II
RESULTS OF THE STATISTICAL COMPARISON STUDY BASED ON THE HYPERVOLUME DIFFERENCE INDICATOR AMONG THE TWO VERSIONS OF D-NSGA-II AND NSGA-II.

			D-NSGA-II		NSGA-II
			W.S.	A.N.	
20 Jobs	$I_{\epsilon+}$	mean	0.4667	0.0526	0.6305
		std	0.0567	0.0014	0.0533
	D-NSGA-II	W.S.	X	$\prec$	$\succ$
		A.N.	$\succ$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X
40 Jobs	$I_{\epsilon+}$	mean	0.1372	0.1687	0.4486
		std	0.0092	0.0215	0.0326
	D-NSGA-II	W.S.	X	$\prec$	$\succ$
		A.N.	$\succ$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X
60 Jobs	I_{HD}	mean	0.0884	0.1611	0.383
		std	0.0064	0.004	0.0388
	D-NSGA-II	W.S.	X	$\succ$	$\succ$
		A.N.	$\prec$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X
80 Jobs	I_{HD}	mean	0.0723	0.1949	0.3691
		std	0.0039	0.0021	0.053
	D-NSGA-II	W.S.	X	$\succ$	$\succ$
		A.N.	$\prec$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X
100 Jobs	I_{HD}	mean	0.1281	0.238	0.4368
		std	0.1281	0.0018	0.0299
	D-NSGA-II	W.S.	X	$\succ$	$\succ$
		A.N.	$\prec$	X	$\succ$
	NSGA-II		$\prec$	$\prec$	X

TABLE III
RESULTS OF THE STATISTICAL COMPARISON STUDY BASED ON THE EPSILON DIFFERENCE INDICATOR AMONG THE TWO VERSIONS OF D-NSGA-II AND NSGA-II.

of points obtained by decoding a partial solution with the Aneja-Nair method.

As a future work it will be interesting to make additional comparison study with other stopping criteria in order to let time to each approach to converge. Also, it will be interesting to apply Aneja-Nair method with a fixed maximum number of iterations n'_α. This way, it will permit to control the number of points decoding a partial solution and to have a better repartition on the Pareto front than with a classical weighted sum method.

Figures 7, 8, 9 and 10 show empirical attainment surface for NSGA-II and D-NSGA-II on instances of 20 and 100 jobs. In the figures the line "median" represents the curve where the fraction of attainable points (that is, dominated in Pareto sense) is 50%. The line "worst" represents points that are always attainable and line "points" the points that are attainable at less once. On both instances D-NSGA-II has a much more bigger empirical attainment surface than the classical NSGA-II. In case of 20 jobs it can be observed that D-NSGA-II(N.A.) has bigger empirical attainment surface than D-NSGA-II(W.S.) but the opposite situation is observed for 100 jobs case.

VI. CONCLUSION

In this paper, an efficient multi-objective decoder-based hybrid method, called D-NSGA-II (Decoder based NSGA-II), has been designed to solve the bi-objective Just-In-Time Job-Shop Scheduling Problem (JIT-JSP). This method is based on a decoding strategy where a partial solution is decoded into a Pareto set of solutions. A solution of bi-objective JIT-JSP is indirectly represented by a permutation order while a

	Weighted Sum	Aneja-Nair Method
20 Jobs	6	9.5
40 Jobs	3	10.1
60 Jobs	4	13.9
80 Jobs	3	14.2
100 Jobs	3	17.8

TABLE IV
AVERAGE NUMBER OF POINTS DECODING A PARTIAL SOLUTION USING WEIGHTED SUM METHOD OR ANEJA-NAIR METHOD.

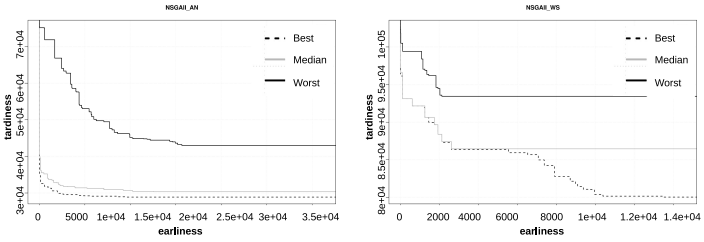

Fig. 7. Empirical Attainment Function of D-NSAGA-II on instance of 20 jobs. With Aneja-Nair based decoder at left and simple weighted sum at right.

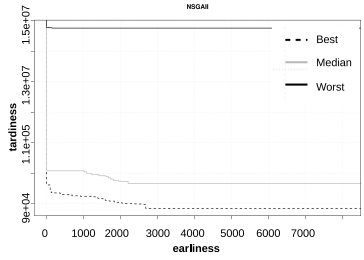

Fig. 8. Empirical Attainment Function of NSAGA-II on instance of 20 jobs.

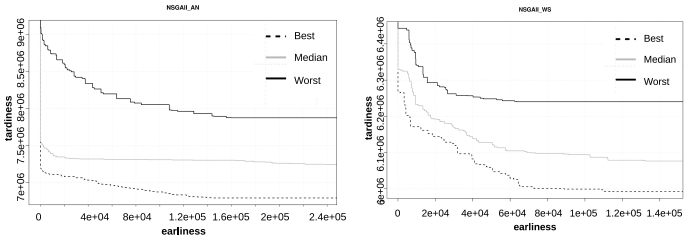

Fig. 9. Empirical Attainment Function of D-NSAGA-II on instance of 100 jobs. With Aneja-Nair based decoder at left and simple weighted sum at right.

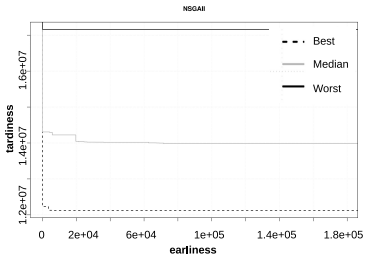

Fig. 10. Empirical Attainment Function of NSAGA-II on instance of 100 jobs.

decoder is used to solve the bi-objective Timing sub-problem and to fix the idle times. Two algorithms have been designed for the decoder of D-NSGA-II and compared. The first one uses the weighted sum method with several predefined weights. The second one uses Aneja-Nair method and is optimal. The comparison study shows that the decoder based on Aneja-Nair method is more efficient on small instances whereas, for big instances, the weighted sum method performs better. All the proposed decoding strategies outperform the classical NSGA-II algorithm.

Moreover, as the biggest drawback of the Aneja-Nair based decoder is computation time, it could be interesting to fix its maximum number of iterations. This will reduce the number of points in the approximation of the Pareto front as it is done for the weighted sum method.

A deeper comparison with different stopping criteria will be used in future work to compare the methods. We believe that an interesting line of research will be to study other evaluation strategy for multi-objective decoder-based hybrid metaheuristic. Particularly, one of our perspective of work is to study indicator based evaluation on this kind of problem.

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[2] F. Sourd and S. Kedad-Sidhoum, "An efficient algorithm for the earliness-tardiness scheduling problem," *Optimisation Online,(1205)*, 2005.

[3] J. Schaller, "A comparison of lower bounds for the single-machine early/tardy problem," *Computers & operations research*, vol. 34, no. 8, pp. 2279–2292, 2007.

[4] S. Tanaka and S. Fujikuma, "A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time," *Journal of Scheduling*, vol. 15, no. 3, pp. 347–361, 2012.

[5] S. Kedad-Sidhoum and F. Sourd, "Fast neighborhood search for the single machine earliness–tardiness scheduling problem," *Computers & Operations Research*, vol. 37, no. 8, pp. 1464–1471, 2010.

[6] L. Liu and H. Zhou, "Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem," *Information Sciences*, vol. 226, pp. 68–92, 2013.

[7] T. Qin, B. Peng, U. Benlic, T. Cheng, Y. Wang, and Z. Lü, "Iterated local search based on multi-type perturbation for single-machine earliness/tardiness scheduling," *Computers & Operations Research*, vol. 61, pp. 81–88, 2015.

[8] A. Rahimi-Vahed, M. Dangchi, H. Rafiei, and E. Salimi, "A novel hybrid multi-objective shuffled frog-leaping algorithm for a bi-criteria permutation flow shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 41, no. 11-12, pp. 1227–1239, 2009.

[9] K. R. Baker and G. D. Scudder, "Sequencing with earliness and tardiness penalties: a review," *Operations research*, vol. 38, no. 1, pp. 22–36, 1990.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[11] M. R. Garey, R. L. Graham, and D. S. Johnson, "Performance guarantees for scheduling algorithms," *Operations research*, vol. 26, no. 1, pp. 3–21, 1978.

[12] J. Bauman and J. Józefowska, "Minimizing the earliness–tardiness costs on a single machine," *Computers & Operations Research*, vol. 33, no. 11, pp. 3219–3230, 2006.

[13] T. Murata and H. Ishibuchi, "Performance evaluation of genetic algorithms for flowshop scheduling problems," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. IEEE, 1994, pp. 812–817.

[14] H. Khorshidian, N. Javadian, M. Zandieh, J. Rezaeian, and K. Rahmani, "A genetic algorithm for jit single machine scheduling with preemption and machine idle time," *Expert systems with applications*, vol. 38, no. 7, pp. 7911–7918, 2011.

[15] M. Mahnam, G. Moslehi, and S. M. T. F. Ghomi, "Single machine scheduling with unequal release times and idle insert for minimizing the sum of maximum earliness and tardiness," *Mathematical and Computer Modelling*, vol. 57, no. 9, pp. 2549–2563, 2013.

[16] R. James and J. T. Buchanan, "A neighbourhood scheme with a compressed solution space for the early/tardy scheduling problem," *European Journal of Operational Research*, vol. 102, no. 3, pp. 513–527, 1997.

[17] M. Ehrgott, *Multicriteria optimization*. Springer Science & Business Media, 2006.

[18] Y. P. Fang, K. Meng, and X. Q. Yang, "Piecewise linear multicriteria programs: the continuous case and its discontinuous generalization," *Operations research*, vol. 60, no. 2, pp. 398–409, 2012.

[19] Y. P. Aneja and K. P. Nair, "Bicriteria transportation problem," *Management Science*, vol. 25, no. 1, pp. 73–78, 1979.

[20] S. Jacquin, L. Moussin, I. Machado, E. Talbi, and L. Jourdan, "A comparison of decoding strategies for the 0/1 multi-objective unit commitment problem," in *Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 381–395.

[21] K. Bülbül, P. Kaminsky, and C. Yano, "Flow shop scheduling with earliness, tardiness, and intermediate inventory holding costs," *Naval Research Logistics (NRL)*, vol. 51, no. 3, pp. 407–445, 2004.

[22] J. Knowles, L. Thiele, and E. Zitzler, "A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, TIK Report 214, Feb. 2006.

[23] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," IRIDIA, Tech. Rep., 2011.

[24] A. Liefooghe, L. Jourdan, and E.-G. Talbi, "A software framework based on a conceptual unified model for evolutionary multiobjective optimization: Paradiseo-moeo," *European Journal of Operational Research*, vol. 209, no. 2, pp. 104–112, 2011.

[25] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "Pisa: A platform and programming language independent interface for search algorithms," in *EMO*, 2003, pp. 494–508.