Applied Soft Computing 13 (2013) 4118–4132

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Solving bi-objective flow shop problem with hybrid path relinking algorithm

Rong-Qiang Zeng [a,b,c,d], Matthieu Basseur [b,*], Jin-Kao Hao [b]

[a] School of Mathematics, Southwest Jiaotong University, Chengdu, Sichuan 610031, PR China
[b] LERIA, Université d'Angers, 2, Boulevard Lavoisier, 49045 Angers Cedex 01, France
[c] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, PR China
[d] Chengdu Documentation and Information Center, Chinese Academy of Sciences, Chengdu, Sichuan 610041, PR China

ARTICLE INFO

Article history:
Received 3 April 2012
Received in revised form 31 January 2013
Accepted 29 May 2013
Available online 19 June 2013

Keywords:
Path relinking
Hypervolume contribution
Multi-objective optimization
Local search
Flow shop problem (FSP)

ABSTRACT

This paper presents and investigates different ways to integrate path relinking techniques into the hypervolume-based multi-objective local search algorithm (HBMOLS). We aim to evaluate the effectiveness of different path relinking strategies, these strategies focus on two main steps: the ways of path generation and the mechanisms of solutions selection. We propose different methods to establish the path relinking algorithms in a multi-objective context. Computational results on a bi-objective flow shop problem (FSP) and a statistical comparison are reported in the paper. In comparison with two versions of HBMOLS, the algorithms selecting a set of solutions located in the middle of the generated path are efficient. The behavior of these algorithms sheds light on ways to further improvements.

1. Introduction

Path Relinking (PR) was originally proposed by Glover [12] as an intensification strategy of exploring trajectories connecting elite solutions obtained by tabu search or scatter search. In contrast with other evolutionary procedures, such as genetic algorithms, PR provides "unifying principles for joining solutions based on generalized path constructions" [13]. The main purpose of the present study is to integrate the path relinking techniques into hypervolume-based multi-objective optimization.

In single objective optimization, a total order relation can be easily used to rank the solutions. However, such a natural total order relation does not exist in multi-objective optimization. Indeed, there usually does not exist one optimal but a set of solutions called Pareto optimal solutions or efficient solutions. The aim is to generate the set of Pareto optimal solutions called the Pareto set, which keeps the best compromise solutions among all the objectives.

We define a hypervolume contribution indicator that assigns a fitness value to each solution. The fitness value is computed according to Pareto dominance relation and the hypervolume contribution principle. Moreover, this indicator induces an order to rank the solutions so that it can be used in the selection process of multi-objective optimization algorithms.

In this paper, we present a study of using path relinking for hypervolume-based bi-objective optimization. In particular, we introduce five different methods to generate a path using a unique neighborhood operator, and we consider four different mechanisms to select a set of solutions. This set will be provided as the entry to the hypervolume-based multi-objective local search algorithm (HBMOLS), which is dedicated to improve the solutions generated by the path relinking algorithm. Therefore, we propose 20 different versions of hybrid algorithm. In order to evaluate the effectiveness of these algorithms, we apply them to a bi-objective flow shop problem, where we aim to minimize two objective functions: the total completion time and the total tardiness.

The rest of this paper is organized as follows: Section 2 is devoted to introduce the definitions of multi-objective optimization, hypervolume contribution indicator and related works. In Section 3, we propose the hybrid multi-objective path relinking algorithms. Then, we apply this algorithm to solve the bi-objective flow shop problem in Section 4. Afterwards, Section 5 presents the computational results and the performance analysis of the algorithms. Finally, the conclusions and perspectives for the future work are discussed in the last section.

* Corresponding author. Tel.: +33 241735294.
E-mail addresses: zeng@info.univ-angers.fr, zrq777@gmail.com (R.-Q. Zeng), basseur@info.univ-angers.fr (M. Basseur), hao@info.univ-angers.fr (J.-K. Hao).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.05.018

2. Preliminaries

In this section, we first present some basic notations and definitions related to the multi-objective optimization. Then, we briefly introduce the hypervolume contribution indicator. Finally, we briefly investigate the literature which is related to the present study.

2.1. Multi-objective optimization

First, let us recall some useful notations and definitions of multi-objective optimization problems (MOPs), which are taken from [29]. Let X denote the search space of the optimization problem under consideration and Z the corresponding objective space. Without loss of generality, we assume that $Z = \Re^n$ and that all n objectives are to be minimized. Each $x \in X$ is assigned exactly one objective vector $z \in Z$ on the basis of a vector function $f: X \to Z$ with $z = f(x)$. The mapping f defines the evaluation of a solution $x \in X$, and often one is interested in those solutions that are Pareto optimal with respect to f. The relation $x_1 \succ x_2$ means that the solution x_1 is *preferable* to x_2. The dominance relation between two solutions x_1 and x_2 is usually defined as follows [29]:

Definition 1. A decision vector x_1 is said to dominate another decision vector x_2 (written as $x_1 \succ x_2$), if $f_i(x_1) \leq f_i(x_2)$ for all $i \in \{1, \ldots, n\}$ and $f_j(x_1) < f_j(x_2)$ for at least one $j \in \{1, \ldots, n\}$.

Definition 2. $x \in X$ is said to be Pareto optimal solution if and only if a solution $x_i \in X$ which *dominates* x does not exist.

Definition 3. *POS* is said to be a Pareto optimal set if and only if *POS* is composed of all the Pareto optimal solutions.

Definition 4. *NDS* is said to be a non-dominated set if and only if any two solutions $(x, y) \in NDS$ which do not dominate each other.

Since in most cases, it is not possible to compute the Pareto optimal set in a reasonable time, we are interested in computing a set of non-dominated solutions which is as close to the Pareto optimal set as possible. Therefore, the whole goal is often to identify a good approximation of the Pareto optimal set.

2.2. Hypervolume contribution indicator

In multi-objective optimization, we have to define a strategy to assign a fitness value to each solution. Currently, a popular metric used to compare the fronts in the objective space is the hypervolume measure (also known as S-metric or the Lebesgue measure). Hypervolume is the n-dimensional space that is dominated by the points (solutions) in the objective space [28]. A front with a larger hypervolume is likely to present a better set of trade-offs to a user than a front with a smaller hypervolume.

Based on the hypervolume measure, we define a Hypervolume Contribution (HC) indicator, which is used to evaluate the contribution of each solution in the objective space [4]. Furthermore, we divide the entire population into two sets, one is a set of non-dominated solutions, the other is a set of solutions dominated by at least one solution in the first set.

More precisely, according to this dominance relation and hypervolume contribution principle, we assign different values to the solutions from these two sets: a negative value is assigned to each solution located in the dominated set. The value corresponds to the greatest area between a non-dominated solution and the considered solution (see Fig. 1). A positive value is assigned to each solution located in the non-dominated set. The value corresponds to the area dominated by the considered solution and not dominated by any other non-dominated solutions (see Fig. 2). The fitness value can be exactly calculated as follows:

$$HC(x, P) = \begin{cases} -(f_1(x) - f_1(y)) \times (f_2(x) - f_2(y)), & \text{if } x \text{ is dominated} \\ (f_1(y_1) - f_1(x)) \times (f_2(y_0) - f_2(x)), & \text{if } x \text{ is non} - \text{dominated} \end{cases} \quad (1)$$

Let us mention that in general the values of the reference point have to be set much bigger (at least twice) than the objective values of all the solutions in the population, which allows us to be sure to keep the extreme non-dominated solutions during the search process (it has no influence on other non-dominated solutions).

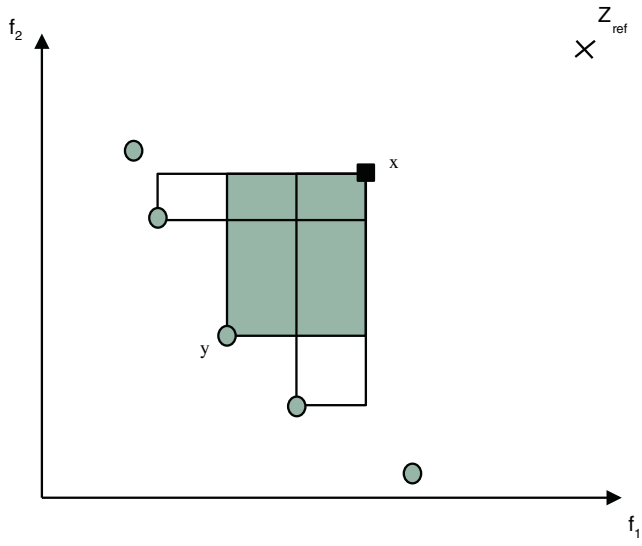

Fig. 1. $HC(x, P)$ of a dominated solution x to a population P: maximum dominance area (grey box) computed between x and y ($y \succ x$, $y \in P$) (Z_{ref}: the reference point).

2.3. Related works

Up to now, many heuristic and metaheuristic methods have been proposed to solve all kinds of combinatorial optimization problems. Some of the most effective methods use the PR principles, where the objective is to explore the search space by creating paths within a given set of high-quality solutions.

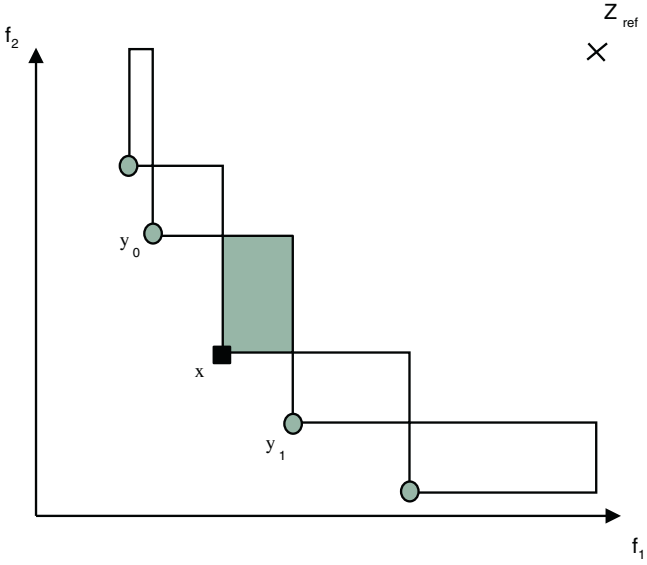

Fig. 2. $HC(x, P)$: hypervolume contribution of a solution x to a population P, the area dominated only by the solution x is colored in grey (y_0 and y_1: two non-dominated solutions next to x, Z_{ref}: the reference point).

In this paper, we aim to solve bi-objective flow shop problem (FSP), which is a class of widely studied problems with strong engineering backgrounds. The FSP schedules with N jobs and M machines, where each job has to be processed on all the machines in the same machine sequence. Each machine could only process one job at a time, and the machines can not be interrupted once they start processing a job. As soon as the operation is finished, the machines become available. In the following, we will focus on the studies dealing with FSP using the PR approaches.

Reeves and Yamada [21] develop a genetic algorithm embedded PR techniques to minimize the makespan of FSP. In their algorithm, they select two solutions with a probability, which is based on a linear ranking of the makespan values. Then, the distance between the two selected solutions is calculated according to the precedence-based measure. In order to generate a path, they define a critical-block based neighborhood, which is a set of moves that shift a job in a critical block to some position in one of the other block. Then, one of the neighbors is accepted with the probability based on the objective values. Experimental results demonstrate the effectiveness of this method.

In [7], Bozejko and Wodecki propose a parallel path-relinking approach for minimizing the makespan and the total flow time of FSP, respectively. According to the principle described in [21], they use Kendall's τ measure as the distance measure to construct a path. The whole algorithm is based on the idea of evaluation of so-called starting solutions set presented in [14]. They report very promising results.

Recently, Vallada et al. [27] present three genetic algorithms for minimizing the total tardiness of FSP. One of these algorithms includes PR techniques, which are used as crossover or applied to the best individuals found so far without improvement after a number of iterations. They propose four different ways to select initial individuals such as two individuals of the current population, two best individuals of the elite pool, two randomly selected individuals from the elite pool and a best individual with a randomly selected individual. Then the path between the two selected individuals is generated by the interchange movements with the swap operator. The results show that this algorithm is very effective.

Costa et al. [10] propose a GRASP with path-relinking for FSP with the criterion of minimizing the total flow time. This algorithm uses a distinct truncated path-relinking operator that interrupts path-relinking as soon as a new solution improving the original one is found. Then, this method resumes the local search phase bypassing the construction phase. In comparison to the classical hybridization strategy between GRASP and path-relinking, this version improves significantly the quality of the produced solutions.

As an excellent search strategy, PR has proved its efficiency in single objective optimization [12]. Actually, these PR methods focus on solving single-objective FSP. It is convenient for these algorithms to construct a path and select the best solution from the path in the single-objective case. However, it is not trivial to integrate PR techniques into multi-objective algorithms. There are only a few studies that propose PR algorithms to solve multi-objective optimization problems. Indeed, we have to reconsider some classical issues in PR principles, such as distance measure, path selection, improvements of the intermediate solutions and so on.

Pasia et al. [20] present three PR approaches for solving bi-objective FSP. By using a straightforward implementation of the ant colony system, they first generated two pools of initial solutions, where one pool contains solutions that are good with respect to the makespan and the other contains solutions that are good with respect to the total tardiness. Based on random insertion, all the solutions in both pools are improved by local search in order to obtain a non-dominated set. Then, two solutions are randomly selected from this set to construct a path. Along the path, some of the solutions are submitted for improvement. The authors propose three different strategies to define the heuristic bounds. Each strategy allows the solutions to undergo local search under the conditions based on local nadir points. However, the heuristic bounds usually limit the search space to a relatively small areas, which has an affection on the effectiveness of local search. Computational results demonstrate that their PR approaches are competitive.

On the other hand, Basseur et al. [6] propose a multi-objective approach to integrate PR techniques into an adaptive genetic algorithm, which is dedicated to obtain a first well diversified approximation of the Pareto set. Based on this set, two solutions are randomly selected to generate a path. According to the distance measure defined in [6], there are many choices from one solution to another one at each step. Then, the authors apply a random aggregation of the objectives to select only one solution from all possible eligible solutions. But this method merely select one solution at random without any comparison with other eligible solutions. After linking these two solutions, a Pareto local search is applied in order to improve the quality of the non-dominated set generated by the PR algorithm. Experimental results on bi-objective FSP show that the PR approach is very promising and efficient.

3. Multi-objective hybrid path relinking algorithm

The Hybrid Path Relinking algorithm (HPR) presented in this section consists in a combination of the Hypervolume-Based Multi-Objective Local Search algorithm (HBMOLS) and a Multi-Objective Path Relinking algorithm (MOPR). The outline of HPR is illustrated in Algorithm 1. In the following sections, we will describe the implementations of the most important steps in HPR.

Algorithm 1. Basic Algorithmic Framework of HPR

```
Input: N (Population size)
Output: A (Pareto approximation set)
Initialization: P ← N randomly generated solutions
                A ← Non-dominated solutions of P
While running time is not reached do
    1) P ← HBMOLS (P)    /* Section 3.1 */
    2) A ← Non-dominated solutions of P ⋃ A
    3) P ← MOPR (A)      /* Section 3.2 */
End while
Return A
```

In HPR algorithm, the individuals of the initial population are generated randomly, i.e., each individual is initialized with a random permutation. Then, each solution in the population is optimized by the HBMOLS algorithm (Section 3.1). Then, we construct a path between two solutions, which are randomly chosen from Pareto approximation set A (MOPR algorithm). After that, some solutions in the path are selected to build a new population P which will be improved by the HBMOLS algorithm (Section 3.2). In other words, path relinking methods can be seen as a way to reinitialize population in the HBMOLS algorithm.

3.1. Hypervolume-based multi-objective local search

As an intensification search strategy, PR needs some efficient initial solutions to generate a set of new solutions in the path. For this purpose, we apply hypervolume-based multi-objective local search [4] to the initial population, which is presented in Algorithm 2.

Algorithm 2. Hypervolume Based Multi-Objective Local Search (HBMOLS)

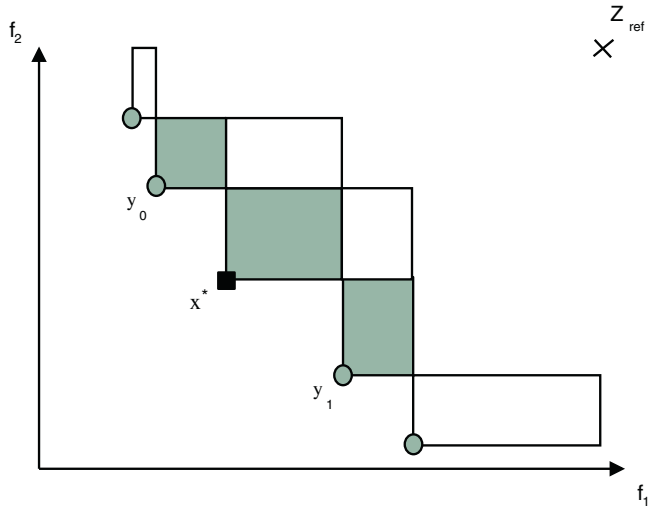

Fig. 3. *HC* fitness update: new non-dominated solution found (grey boxes: new dominance areas to compute).

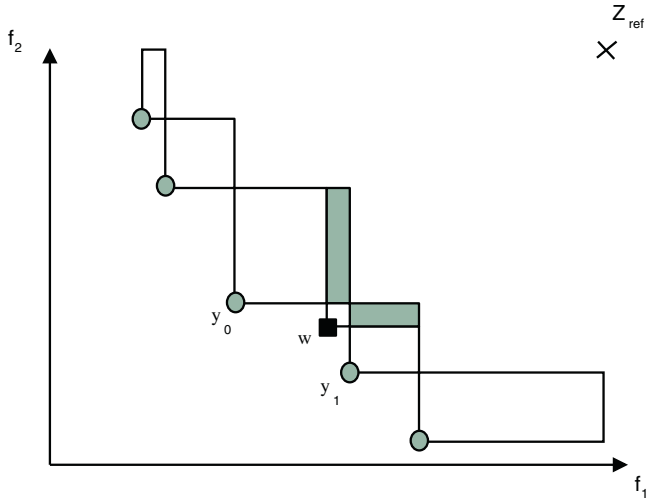

Fig. 4. *HC* fitness update: deletion of a non-dominated solution. The fitness values of solution y_0 and y_1 are computed by adding corresponding grey area respectively (y_0 and y_1: two non-dominated solutions next to w).

1: **Input**: P (Initial population)
2: **Output**: A: (Pareto approximation set)
3: **Local Search Step**:
1) Fitness Assignment: Calculate a fitness value for each $x \in P$, i.e., $Fit(x) \leftarrow HC(x, P)$
2) **For** each $x \in P$ **do**:
repeat
a) $x^* \leftarrow$ one randomly chosen unexplored neighbors of x
b) Progress $\leftarrow$ **Hypervolume Contribution Selection** (P, x^*)
until all neighbors are explored or Progress = True

Algorithm 3. Hypervolume Contribution Selection

Step:
1) $P \leftarrow P \bigcup x^*$
2) compute x^* fitness: $HC(x^*, P)$, then update all $z \in P$ fitness values: $Fit(z) \leftarrow HC(z, P)$
3) $w \leftarrow$ worst individual in P
4) $P \leftarrow P \backslash \{w\}$, then update all $z \in P$ fitness values: $Fit(z) \leftarrow HC(z, P \backslash \{w\})$
5) **if** $w \neq x^*$, **return** True

Algorithm 2 shows the pseudo-code of the HBMOLS procedure. In this procedure, according to Eq. (1), a fitness value is assigned to the solution x^*, which is one of the unexplored neighbors of x. If x^* is dominated, the fitness values of solutions in the non-dominated set are not modified; if x^* is non-dominated, the fitness values of some solutions, which are the neighbors of x^* in the objective space, need to be updated. An example of this case is illustrated in Fig. 3, where the fitness values of two neighbors y_0 and y_1 of x^* are updated by Eqs. (2) and (3), respectively.

$$HC(y_0, P) = HC(y_0, P \backslash \{x^*\}) \times \frac{f_1(x^*) - f_1(y_0)}{f_1(y_1) - f_1(y_0)} \quad (2)$$

$$HC(y_1, P) = HC(y_1, P \backslash \{x^*\}) \times \frac{f_2(x^*) - f_2(y_1)}{f_2(y_0) - f_2(y_1)} \quad (3)$$

Then, the solution with the worst fitness value is removed from the population P. If this solution is dominated, nothing will be changed in the non-dominated set; otherwise, we should update the fitness values of the neighbors of this solution in the objective space. In Fig. 4, the solution ω is discarded, which has the smallest fitness value. For this example, the fitness values of solutions y_0 and y_1 are updated by Eqs. (4) and (5).

$$HC(y_0, P \backslash \{\omega\}) = HC(y_0, P) \times \frac{f_1(y_1) - f_1(y_0)}{f_1(\omega) - f_1(y_0)} \quad (4)$$

$$HC(y_1, P \backslash \{\omega\}) = HC(y_1, P) \times \frac{f_2(y_0) - f_2(y_1)}{f_2(\omega) - f_2(y_1)} \quad (5)$$

The whole procedure will repeat until the termination criterion is satisfied. By this way, the quality of the entire population is improved by using the HC indicator as the selection measure.

3.2. Multi-objective path relinking algorithm

The outline of the multi-objective path relinking algorithm is presented in Algorithm 4. In PR algorithms, two solutions used to generate a path are usually called an initial solution and a guiding solution, respectively. These solutions can be randomly selected from a population of "good" solutions, different mechanisms can be used to select these solutions. In particular, we can select them from different sets, and the number of these solutions should be at least 2. Then a path is generated according to a set of solutions [12].

Algorithm 4. Multi-Objective Path Relinking (MOPR)

1: **Input**: A (Pareto approximation set)
2: **Output**: P (For HBMOLS execution)
3: **Path Relinking**:
1) $P \leftarrow N$ randomly generated solutions
2) randomly choose initial solution x_i and guiding solution x_j from A
3) compute the distance d_{ij} between x_i and x_j
4) generate a set of solutions $T = \{t_1, t_2, \cdots, t_{d_{ij}-1}\}$ along the path between x_i and x_j
5) select n_{pr} solutions from the set T
6) **for** $i \leftarrow 1, \ldots, n_{pr}$ **do**
4: **Hypervolume Contribution Selection** (P, y_i)
5: **end for**
6: **return** P

In this paper, we will focus on the basic strategies of path generation by using one initial solution and one guiding solution. Moreover, the initial solution and the guiding solution are selected from a single set, i.e., the Pareto approximation set A obtained after applying HBMOLS to the population P. The initial solution i and the guiding solution j are uniformly randomly selected from A (see Fig. 5).

In order to generate a path, we define a distance between two solutions. With the distance measure, we define a minimal path whose length is equal to the value of distance. In addition, there are many choices for path construction between two solutions, we generate one path among the possible paths of minimal length. At each step, we generate only one new solution and make sure the distance

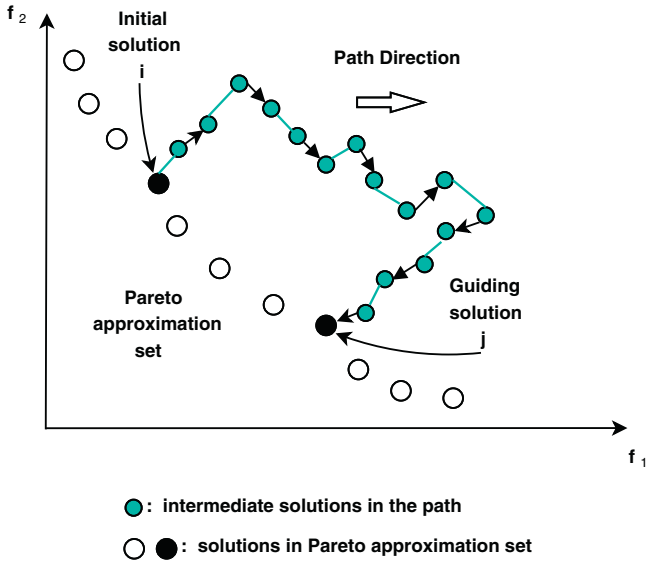

Fig. 5. MOPR: path generation between an initial solution i and a guiding solution j, which are randomly chosen from a Pareto approximation set.

between the new solution and the guiding solution decreases by 1. This is the main part of the MOPR algorithm. After the path generation, a subset of solutions belonging in the path are selected to return P, which initially contains the randomly generated solutions. Finally, each selected solution is assigned a fitness value by the HC indicator and compared with the solutions in the population P. The solution with the worst fitness value will be deleted from the population. Therefore, the output P is composed of some random solutions and some solutions selected from the path.

4. Application to bi-objective flow shop problem

The Flow Shop Problem (FSP) is one of the most thoroughly studied machine scheduling problems, which schedules a set of jobs on a set of machines according to a specific order. Many optimization algorithms have been proposed to solve this problem.

In [16], Lemesre et al. propose a parallel exact method, which is a very general scheme to optimally solve bi-objective combinatorial optimization problems. Then, an effective multi-objective optimization algorithm is proposed in [1], which is based on a Multi-objective Immune System (MOIS). Recently, Khalili and T. Moghaddam [15] present a new multi-objective electromagnetism algorithm, according to the attraction–repulsion mechanism of electromagnetic theories. While in [18], A. Liefooghe et al. propose an indicator-based evolutionary algorithm with four different strategies which are based on uncertainty-handling quality indicators.

In this section, we apply the hybrid path relinking algorithm to solve bi-objective flow shop problem. First, we give the definition of the bi-objective FSP and introduce the insertion operator used to explore the neighborhood of a solution. Then, we present the distance measure used to construct a path and discuss the exact steps of path generation and solution selection.

4.1. Problem definition

The bi-objective FSP can be presented as a set of N jobs $\{J_1, J_2, \ldots, J_n\}$ to be scheduled on M machines $\{M_1, M_2, \ldots, M_m\}$. An example of bi-objective FSP is illustrated in Fig. 6. Machines are critical resources, i.e., one machine can not be assigned to two jobs simultaneously. Each job J_i is composed of m consecutive tasks $t_{i1}, t_{i2}, \ldots, t_{im}$, where t_{ij} represents the jth task of the job J_i requiring the machine m_j. To each task, t_{ij} is associated with a processing time p_{ij}. Each job J_i has a due date d_i. Then, we aim to minimize two objectives [5]: C_{max} (Makespan: Total completion time) and T (Total tardiness).

The task t_{ij} is scheduled at the time s_{ij}. The two objectives can be computed as follows [5]:

$$f_1 = C_{max} = \max_{i\in[1\ldots n]} \{s_{im} + p_{im}\} \tag{6}$$

$$f_2 = T = \sum_{i=1}^{n} \max(0, s_{im} + p_{im} - d_i) \tag{7}$$

C_{max} minimization has been proved to be NP-hard for more than two machines [17] and total tardiness minimization T has been also proved to be NP-hard even with one machine [11].

4.2. The insertion operator

Generally, a candidate solution to bi-objective FSP can be encoded as a permutation π composed of $\{0, \ldots, n-1\}$ values, such that $\pi(i)$ denotes the job to be executed at $(i+1)$th position. As suggested in [6], we employ the insertion operator to π, which consists in inserting a selected job to a designated position.

The insertion neighborhood $N(\pi)$ of a permutation π of jobs is defined by considering all possible pairs of positions $(j, k) \in \{0, \ldots, n-1\}^2$ of π (with $j \neq k$), where the job at position j is removed from π and inserted at position k [22]. The sequence that results from such a move is if $j<k$ (see Part A in Fig. 7), or $\pi(k+1), \ldots, \pi(j-1), \pi(j+1), \ldots, \pi(n-1))$ if $j>k$ [22] (see Part B in Fig. 7).

The size of $N(\pi)$ is equal to $(n-1)(n-1)$. Equivalently, a neighboring solution of $N(\pi)$ can be obtained by applying the insertion operator to π.

4.3. Distance measure

In order to generate a path to link two solutions, we have to choose a measure to compute the distance between these two solutions. Several measures that are able to calculate a distance between two given permutations are discussed in [24]. In [20], Pasia et al. choose Hamming distance as the distance measure to guide the path construction.

Among many types of moves considered in the literature for the FSP, two of them appear prominently: the swap operator and the insertion operator [19]. Taillard [25] showed that the insertion operator is more effective than the swap operator when used in a neighborhood search. Moreover, the insertion operator is also shown to be very efficient to solve multi-objective FSPs in [8].

Therefore, we decide to define our distance measure directly related to the insertion operator. This allow us to know the minimum number of moves, which have to be applied on the initial solution to reach the guiding solution. Correspondingly, this distance depends on the length of the longest common subsequence (LCS) that is shared by two solutions.

Without loss of generality, we consider that both initial and guiding solutions are permutations of the set $\{0, \ldots, n-1\}$. The length of the longest common subsequence between two solutions varies in the interval $[1, n]$. As explained in [6], the distance is defined as n minus the length of LCS, which corresponds to the minimal number of moves we have to apply to link these solutions, and implies that the minimal path between two solutions is always smaller than n. Then, the distance lies between 0 and n. The longest common subsequence can be calculated in $O(n^2)$ by a dynamic programming algorithm, which is similar to the well known Needleman-Wunsch algorithm [9,23]. Besides, the jobs, which do not belong to the LCS, are called the candidate jobs.

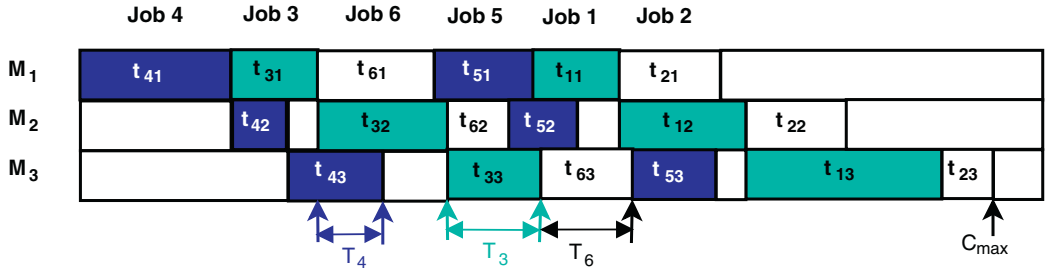

Fig. 6. An example of solution to the bi-objective flow shop problem (6 jobs scheduled on 3 machines). The permutation {4, 3, 6, 5, 1, 2} is associated to this solution.

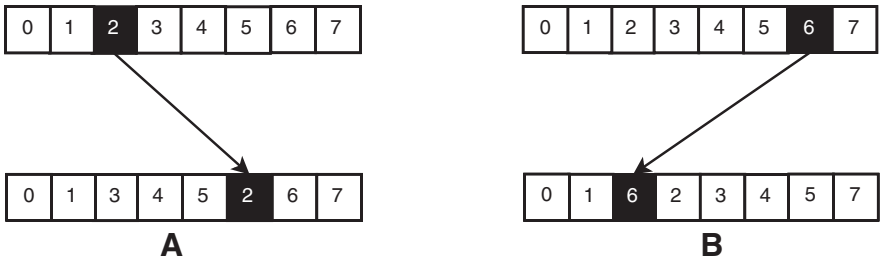

Fig. 7. Two examples of the insertion operator applied on a permutation {0, 1, . . ., 7}. (A) The schedule of job 2 is delayed, the jobs 3, 4, 5, 6 and 7 are shifted to be scheduled earlier. (B) The job 6 is scheduled earlier, the jobs 2, 3, 4 and 5 are shifted to be scheduled later.

An example of the distance between two solutions x and y obtained by computing the LCS is shown in Fig. 8. In this example, the LCS is colored in black, i.e., jobs 14, 18, 0, 1, 5, 8, 3, 11, 9, 10 and 19. The remaining jobs are candidate jobs, i.e., jobs 17, 2, 15, 7, 16, 6, 4, 13 and 12. These jobs will be moved in order to reach the guiding solution y. Additionally, the LCS is not unique in most cases, only one of them is randomly chosen to generate the path.

We denote the problem size as n and the length of the longest common subsequence as l. The distance d between x and y equals to $n-l$. Then, d corresponds to the minimal number of moves which have to be applied on the initial solution to reach the guiding solution (proved in [6]). The PR algorithm aims to link these solutions by generating intermediate solutions in $d-1$ steps. At each step, we generate only one new solution and makes sure that the distance between the new solution and guiding solution decreases by 1. Then there will be exactly $d-1$ new solutions in the path, which corresponds to a minimal path between the initial and guiding solutions. Furthermore, there are usually several possible insertion points for each job. We only select one of them for path generation.

Note that the insertion point refers to the position between two neighboring jobs $\pi(i)$ and $\pi(j)$ of LCS in a permutation, which is denoted as $(\pi(i)\pi(j))$. An example is illustrated in Fig. 9, there exists only one insertion point (83) for the candidate job 6. This point is located between two neighboring jobs 8 and 3 in the initial solution. Moreover, there exists several insertion points between two jobs that are not next to each other. For example, there are two insertion points for the candidate job 17, which are located between the jobs 10 and 19: (10 12) and (12 19). Let us remind that, at the beginning or at the end of the permutation, the insertion point is located before the first job or after the last job. In this case, there is only one insertion point denoted as $(\pi(0))$ or $(\pi(n-1))$. In Fig. 9, the insertion point (14) is unique for the candidate job 16.

4.4. Path generation

After computing the longest common subsequence between an initial solution x and a guiding solution y, we can generate the path in several ways. According to the position of the candidate jobs and their corresponding insertion points, we propose five methods to generate a new intermediate solution.

4.4.1. First insertion | last insertion | random insertion

The first three methods (first insertion, last insertion and random insertion) generates only one solution at each step of the path generation. Then, there is no comparison among all the candidate moves as realized in the two methods presented later. Actually, these three methods differ on which job is selected and where it is inserted.

Considering the first insertion, a new solution is generated by choosing the firstly scheduled candidate job among all the candidate jobs in the initial solution and inserting the selected job into the first possible insertion point in the schedule (see Fig. 10).

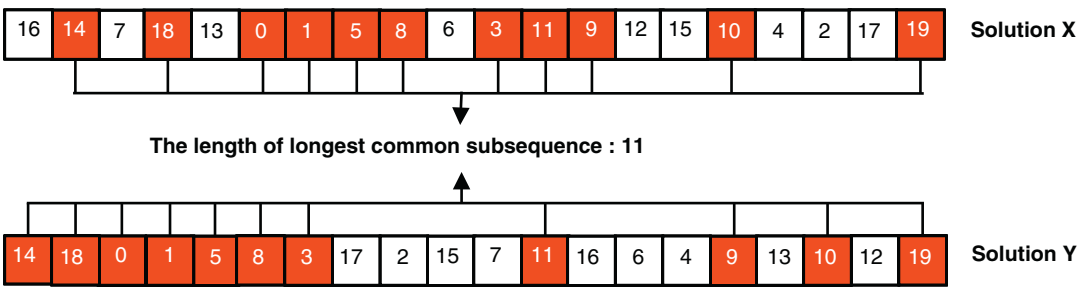

Fig. 8. The longest common subsequence of two permutations x and y. Distance associated: 20 – 11 = 9 (the size of the permutation minus the size of the longest common subsequence).

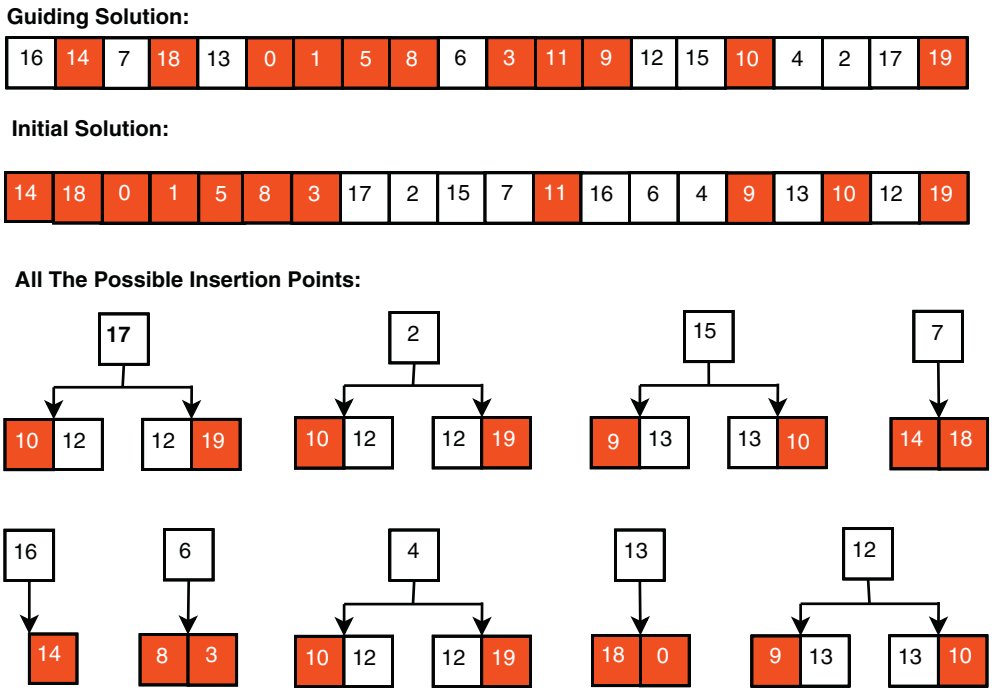

Fig. 9. Insertion points of solution x in different cases.

In this example, the LCS between the initial and guiding solutions is colored in black, the length of which equals to 11. Then, the distance is 9. The job 17 initially equals to the first candidate job in the permutation. Since the job 17 is located between the jobs 10 and 19 in the guiding solution, the job 17 is moved between these two solutions to make sure that the distance between the new solution and guiding solution decreases by 1. We select the first insertion point from two possible points to insert the job 17. After the move, the distance from new solution to the guiding solution equals to 8.

On the contrary, for the last insertion, we select the last candidate job from all the candidate jobs and the last insertion point from all possible insertion points to generate a new solution.

In Fig. 11, we present an example of the last insertion. We compute the LCS (colored in black) between the initial and guiding solutions as we do in the first insertion. Then, we select the last candidate job 12 from the permutation, and insert this job into the last insertion point. After the insertion, we obtain a new solution.

In the random insertion (see Fig. 12), The candidate job and the insertion point are both randomly selected among the candidate moves for path generation. For example, the job 15 and the first insertion point are randomly selected in Fig. 12. Then a new solution is generated.

For these methods, the new solution in the path will replace the initial solution. The process of path generation continue until the distance between the initial solution and the guiding solution reaches 0. Besides, the time complexity of the three methods to generate a new intermediate solution is $O(n^2)$.

4.4.2. *Pareto-based selection and hypervolume-based selection*

The other two path generation methods are Pareto-based selection and hypervolume-based selection, which are slightly different from the first three methods mentioned above. Indeed, all the candidate moves are considered and compared before selecting one of them by using Pareto dominance relation or hypervolume contribution indicator.

An example is shown in Fig. 9, where the jobs colored in black belong to the longest common subsequence. Since the size of the longest common subsequence is 11, we have 9 candidate jobs in the initial solution. For each candidate job, there are usually several possible insertion points (at least 1). For each insertion point,

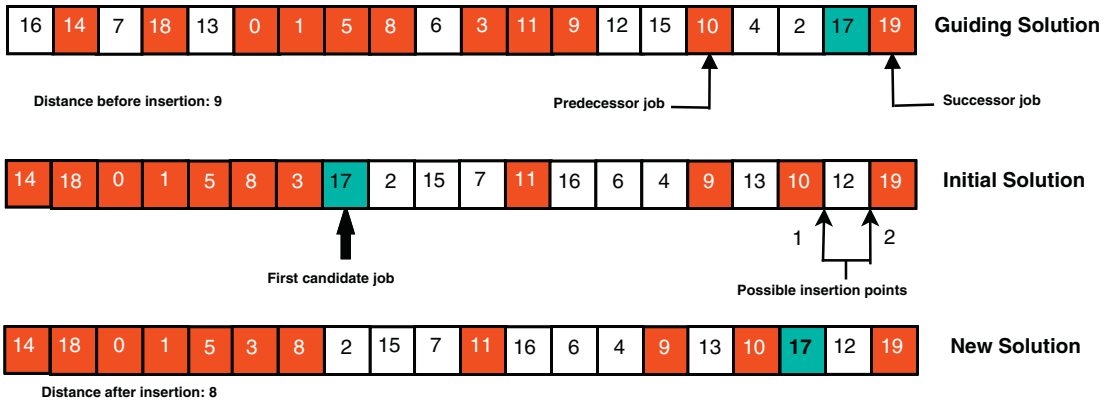

Fig. 10. The first insertion.

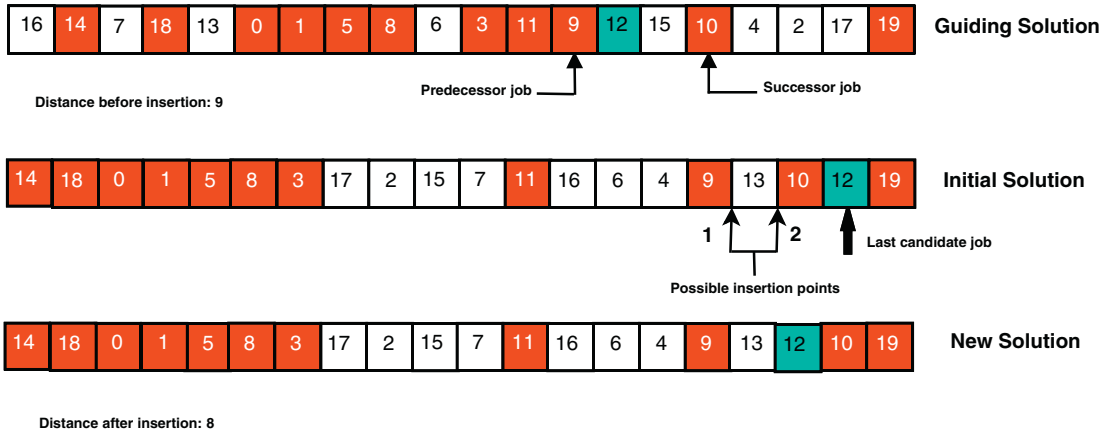

Fig. 11. The last insertion.

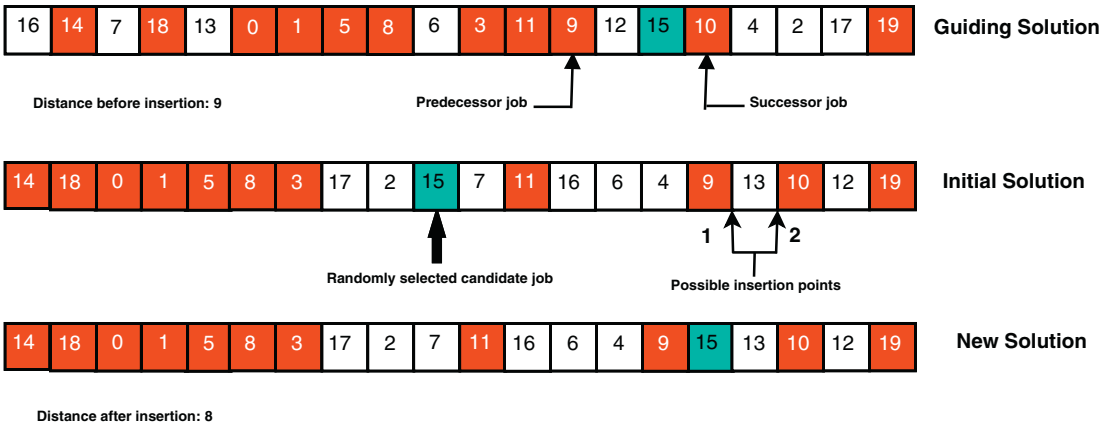

Fig. 12. The random insertion.

we can generate one new solution. For this example, we obtain 14 different solutions. All the possible cases are summarized in Table 1.

Then, we have to define a strategy to evaluate the quality of each possible solution in order to retain the best one. Two methods are illustrated in Fig. 13. In this example, we assume that four solutions (the solutions *A*, *B*, *C* and *D*) belong in the non-dominated set, and the other 10 solutions are dominated by at least one solution in the non-dominated set. Considering the Pareto dominance relation among the candidate solutions, we apply a random selection to the non-dominated solutions, i.e., *A*, *B*, *C* and *D*. In Fig. 13 (left hand side), the solution B is selected.

Alteratively, we can assign a fitness value to each solution by calculating their corresponding hypervolume contribution, and select the solution with the greatesr fitness value as the new solution. Actually, we only need to consider the non-dominated set, since the fitness values of the dominated solutions are negative.

Table 1
Possible moves starting from an initial solution.

	The candidate job in initial solution				
	17	2	15	7	16
Insertion point	(10 12) (12 19)	(10 12) (12 19)	(9 13) (13 10)	(14 18)	(14)

	The candidate job in initial solution			
	6	4	13	12
Insertion point	(5 3)	(10 12) (12 19)	(18 1)	(9 13) (13 10)

Table 2
Parameter values used for the bi-objective FSP instances (*i_j_k* represents the *k*th bi-objective instance with *i* jobs and *j* machines): population size (*N*) and running time (*T*).

Instance	Dim	*N*	*T*	Instance	Dim	*N*	*T*
20_05_01_ta001	20 × 5	10	40″	50_15_01	50 × 15	20	12′30″
20_10_01_ta011	20 × 10	10	80″	50_20_01_ta051	50 × 20	30	16′40″
20_15_01	20 × 15	10	2′	70_05_01	70 × 5	10	8′10″
20_20_01_ta021	20 × 20	10	2′40″	70_10_01	70 × 10	20	16′20″
30_05_01	30 × 5	10	1′30″	70_15_01	70 × 15	30	24′30″
30_10_01	30 × 10	10	3′	70_20_01	70 × 20	30	32′40″
30_15_01	30 × 15	10	4′30″	100_05_01_ta061	100 × 5	20	16′40″
30_20_01	30 × 20	20	6′	100_10_01_ta071	100 × 10	30	33′20″
50_05_01_ta031	50 × 5	10	4′10″	100_15_01	100 × 15	30	50′
50_10_01_ta041	50 × 10	20	8′20″	100_20_01_ta081	100 × 20	40	66′40″

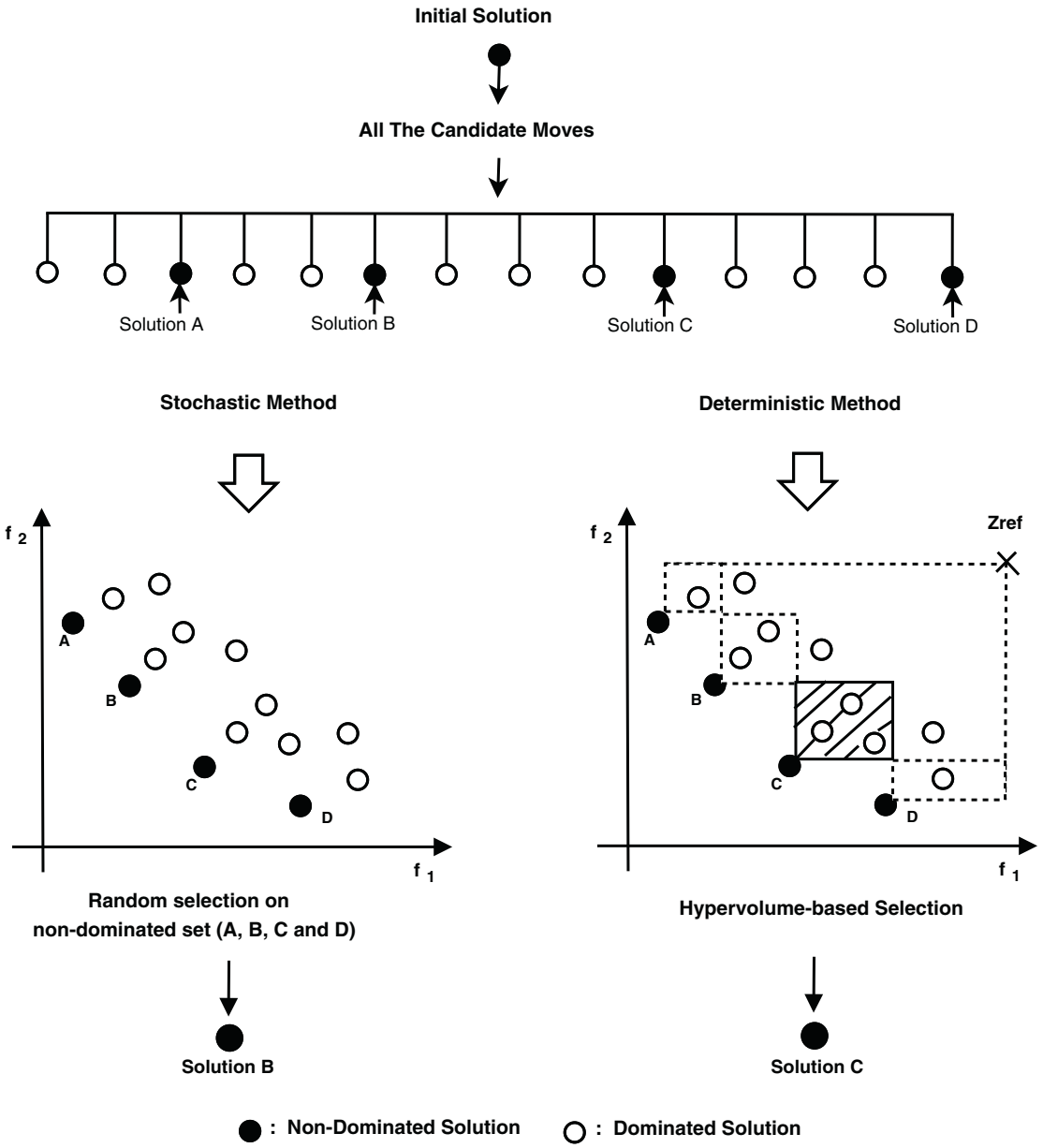

Fig. 13. Two selection methods are considered: In Pareto-Based selection (left hand side), a solution is randomly selected among the non-dominated solutions. In hypervolume-Based selection (right hand side), the solution with the greatest fitness value computed by hypervolume contribution indicator is selected.

Therefore, in Fig. 13 (right hand side), the solution C, with the greatest contribution (the shaded area), is selected.

For the two methods, the new selected solution will also replace the initial solution. The whole process will continue until the distance between the initial solution and the guiding solution becomes 0. Since the comparison is done before generating a new intermediate solution, the time complexity of the two methods is $O(n^3)$.

4.5. Solution selection

After the generation of all the intermediate solutions in the path, we select a set of solutions from this path, which are used to initialize a new population P. These solutions are introduced in P, and we apply the HC indicator to P as the selection measure. To achieve this goal, we propose four mechanisms to select a set of solutions from the generated path. These mechanisms are illustrated in Fig. 14 and described in detail below.

All: All the solutions in the path are selected to be inserted into the population P.

Best: The solutions in the path are divided into two sets, according to their Pareto dominance relations. The solutions belonging to the non-dominated set are selected. In Fig. 14, the solutions A, B, C, D, E and F are selected, which are considered to be in the non-dominated set.

Middle: The solutions located at the beginning or at the end of the path are similar to the initial solution or the guiding solution. since HBMOLS will search the explored areas alike, these solutions could be not very useful. One way to avoid this problem is to select only one solution, which is located in the middle of the path (solution M illustrated in Fig. 14).

***K*-Middle:** Here, we also aim to avoid the problem of proximity of the intermediate solutions to the initial and guiding solutions. Then, we propose to select a set of solutions located in the middle of the path. The number N_{KM} of these solutions is defined according to the length of the generated path. We define this number by

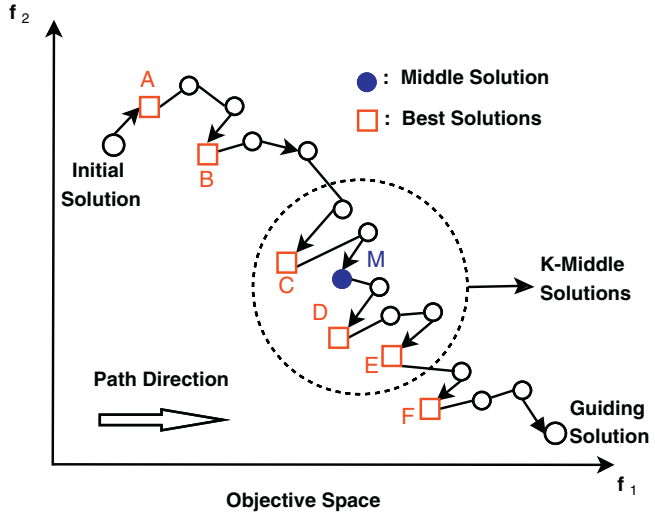

Fig. 14. The mechanisms of solution selection.

using the formula $N_{KM} = |\sqrt{N_{All}}|$, where N_{All} being the number of the solutions in the path, and N_{KM} is the greatest integer that is no greater than $|\sqrt{N_{All}}|$ (see Fig. 14).

5. Experiments

The proposed algorithms are tested on 20 instances, which are taken from Taillard benchmark instances and extended into bi-objective case [26]. Our program is coded in C and compiled by Dev-C++ version 2 compiler. Computational runs were performed on an Intel Core 2 5000B (2 × 2.61 GHz) machine with 2.00 GB RAM.

5.1. Parameter settings

The implementation of the HPR algorithm requires to set a few parameters. In this section, we mainly discuss the running time and the population size.

Running time: The running time T is a key parameter in experiments. We define the time T for each instance by Eq. (8), in which N_{Job} and N_{Mac} represent the number of jobs and the number of machines of an instance, respectively (see Table 2).

$$T = \frac{{N_{Job}}^2 \times N_{Mac}}{50} \text{ s} \tag{8}$$

T is defined according to the "difficulty" of the instance. Indeed, N_{Job} defines the size of the search space, since it is of size N_{Job} !. The roughness of the landscape is strongly related with N_{Mac}. We use this formula to obtain a good balance between the problem difficulty and the time allowed.

Population size: Experiments realized previously on the IBMOLS algorithm showed that the best results are achieved with a small population size N. Taking these results obtained in [3] into account, we define the size of population based on the size of instance. Here, we set this size from 10 to 40 individuals according to Eq. (9), relative to the size of the instance tested (see Table 2).

$$|N| = \begin{cases} 10 & 0 < |N_{Job} \times N_{Mac}| < 500 \\ 20 & 500 \le |N_{Job} \times N_{Mac}| < 1000 \\ 30 & 1000 \le |N_{Job} \times N_{Mac}| < 2000 \\ 40 & 2000 \le |N_{Job} \times N_{Mac}| < 3000 \end{cases} \tag{9}$$

Lastly, there are also other parameters directly related to the bi-objective FSP: the individual coding (permutation of jobs on the first machine and jobs have to be scheduled in the same order on all machines) and the neighborhood operator (insertion operator) [2], etc.

5.2. Performance assessment protocol

With the set of experiments performed, we aim to show the efficiency of our HPR algorithms, which integrate the PR techniques into HBMOLS.

The quality assessment protocol works as follows: we first create a set of 20 runs with different random seeds, different initial populations for each version of the HPR algorithm and each benchmark instance.

To evaluate the quality of k different sets $A_0 \dots A_{k-1}$ of non-dominated solutions obtained on a problem instance, we first compute the set PO^*, which corresponds to the set of non-dominated solutions extracted from the union of all solutions obtained by the different executions. Moreover, we define a reference point $z = [w_1, w_2]$, where w_1 and w_2 correspond to the worst value of each objective function in $A_0 \cup \dots \cup A_{k-1}$. Then, to evaluate a set A_i of solutions, we compute the difference between A_i and PO^* in terms of hypervolume [30]. This hypervolume difference has to be as close to zero as possible.

5.3. Experimental results

In [29], Zitzler and Kuenzli proposed the Indicator-Based Evolutionary Algorithm (IBEA), which is more effective than the classical multi-objective optimization algorithms such as NSGA-II and SPEA2. Furthermore, based on IBEA, the Indicator-Based Multi-Objective Local Search (IBMOLS) algorithm and HBMOLS are proposed in [5] and in [4], respectively. The computational results indicate these two algorithms are superior to IBEA.

In this section, we concentrate on comparing the effectiveness of the HPR algorithm with HBMOLS. We discuss the experimental results, which are obtained by comparing 20 versions of the HPR algorithm with two versions of the HBMOLS algorithm using random mutations and a crossover operator as the initialization mechanism [3]. These two initialization mechanisms works as follows:

Random Mutations: This method is to apply random mutations on the population P. If the size of P is smaller than N, all solutions in P are selected and the missing individuals are also filled with the random individuals. A predefined number of random moves are applied to each selected solution. This number of moves α is related to the size of the problem instance considered (30% of the problem size).

Crossover Operator: This method is to apply a two-point crossover operator used in [5] to P. If the size of P is greater than $2N$, the parents are randomly selected, each individual being selected once at most. If the size of P is smaller than $2N$, all solutions in P are selected and the missing parents are filled with the random individuals.

In the following paragraphs, the names of these two algorithms above are abbreviated to RM and CO, respectively. In order to denote 20 versions of the HPR algorithms briefly and clearly, we provide a short name to each version, according to the combination of the methods of path generation and the mechanisms of solution selection. For instance, "F_B" refers to a multi-objective path relinking algorithm using the first insertion (F) to generate a path and selecting the best solutions (B) from the generated path to update the population P.

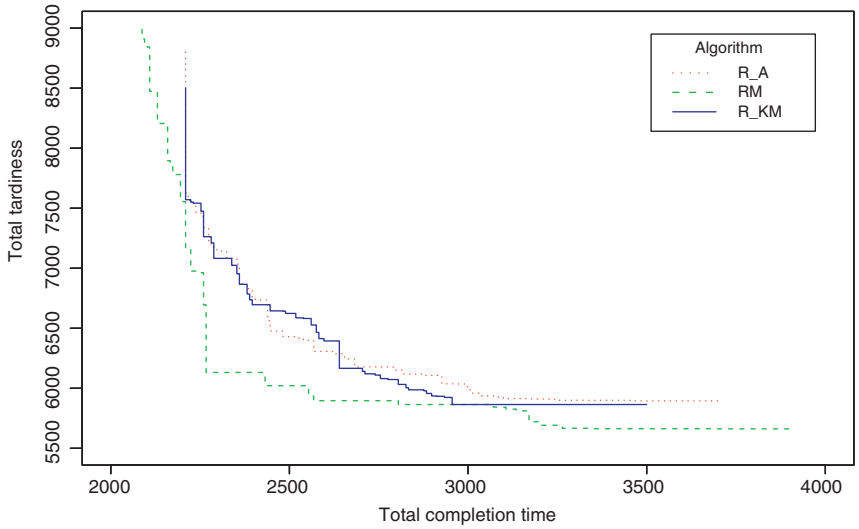

Fig. 15. Empirical attainment functions obtained by the algorithms RM, R_KM and R_A on the instance 30_10_01. The lines correspond to the limit of the objective space which are attained by at least 95% of the runs.

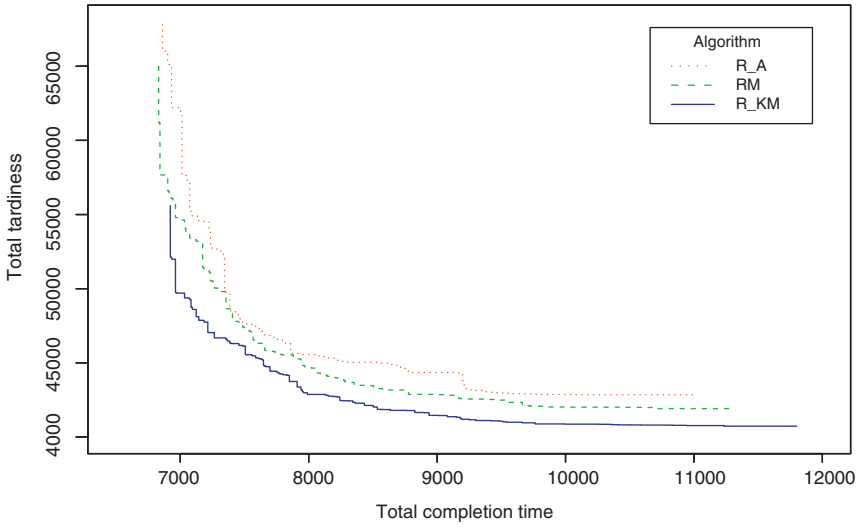

Fig. 16. Empirical attainment functions obtained by the algorithms RM, R_KM and R_A on the instance 100_15_01. The lines correspond to the limit of the objective space which are attained by at least 95% of the runs.

The short names are presented in Table 3. The first column indicates the mechanisms of solution selection. The other five columns correspond to five methods of path generation. More specifically, First, Last, Random, Pareto and HC represent the first insertion, the last insertion, the random insertion, Pareto-based selection and Hypervolume-based selection, respectively.

In Tables 4 and 5, each column contains a value **in grey**, which corresponds to the best result (average hypervolume difference) obtained on the considered instance. The values both **in italic** and **bold** mean that the corresponding algorithms are **not** statistically outperformed by the algorithm which obtains the best result (with a confidence level greater than 95%).

From these two tables, we observe that CO is outperformed by many versions of HPR except on the instance 100_5_01. Furthermore, RM is not statistically outperformed by any other version of HPR on the first seven instances from 20_5_01 to 30_15_01 (see Table 4), and it obtains the best results on the instances 20_5_01, 30_10_01 and 30_15_01. Moreover, the best results are respectively obtained by the *K*-Middle versions of HPR (F_KM, L_KM, R_KM, P_KM and H_KM) on the remaining instances from 30_20_01 to 100_20_01, except the instances 50_10_01 and 50_20_01 (see Tables 4 and 5).

However, there is no version of HPR that statistically outperforms CO and RM on all the instances. Actually, PR techniques have limited contribution on the small instances from 20_5_01 to 30_15_01. We suppose that, since the size of the instance is small, the length of the path is so short that it is difficult to find a set of solutions far enough from the initial and guiding solutions to initiate a new local search. It is more useful to perform the random moves in the search space as done in RM. When we consider the instances with more than 30 jobs, the length of the path becomes

Table 3
The short names of 20 versions of the HPR algorithm.

Solution selection	Path generation				
	First (F)	Last (L)	Random (R)	Pareto (P)	HC (H)
Best (B)	F_B	L_B	R_B	P_B	H_B
Middle (M)	F_M	L_M	R_M	P_M	H_M
All (A)	F_A	L_A	R_A	P_A	H_A
K-Middle (KM)	F_KM	L_KM	R_KM	P_KM	H_KM

Table 4
Comparison of 20 multi-objective path relinking algorithms with hypervolume-based multi-objective local search algorithm (random mutations and crossover operator) on the instances from 20_5_01 to 50_10_01. Each value in the table represents the average hypervolume difference. The first column represents the initialization mechanism. The second and third columns provide the information of solution selection and path generation. The fourth column gives a short name for each algorithm.

Initial	Selection	Path	Name	Instance									
				20_5_01	20_10_01	20_15_01	20_20_01	30_5_01	30_10_01	30_15_01	30_20_01	50_5_01	50_10_01
		First	F_B	**0. 000211**	**0. 000555**	0.004291	0.000253	0.015537	0.054130	**0. 028511**	0.032835	**0.027055**	0.095448
		Last	L_B	**0. 000220**	**0. 000862**	0.006159	0.000269	0.015760	**0. 050603**	0.035292	**0.030072**	0.028242	0.087361
	Best	Random	R_B	0.076627	0.055498	0.073174	0.034508	0.081154	0.200979	0.096203	0.064844	0.083466	0.149919
		Pareto	P_B	**0. 000273**	0.000884	0.005400	0.019619	0.015412	0.059778	0.036944	0.034639	**0.026580**	**0.082310**
		HC	H_B	**0. 000292**	**0. 000753**	0.007592	0.000280	0.016292	0.056763	0.039993	0.039691	0.031269	0.089038
		First	F_M	**0. 000314**	**0. 000607**	0.004617	0.000685	**0.010396**	0.061053	0.042336	0.039683	0.031509	0.108837
		Last	L_M	**0. 000320**	0.000653	0.003256	0.000458	0.013992	0.055309	0.039738	0.040775	0.035707	0.119737
	Middle	Random	R_M	0.093801	0.048349	0.070448	0.024761	0.099705	0.176367	0.105293	0.071167	0.090345	0.132192
		Pareto	P_M	0.000353	**0. 000729**	0.005291	0.000608	0.021271	0.085571	0.050148	0.047683	0.042877	0.130032
		HC	H_M	0.000321	**0. 000727**	0.004048	0.000930	0.020583	0.066211	0.049406	0.047496	0.048463	0.119298
MOPR		First	F_A	0.000304	**0. 001084**	0.007625	0.000235	0.020266	0.056187	0.039879	0.032373	0.028378	**0.077582**
		Last	L_A	0.000335	**0. 000921**	0.005114	0.000181	0.026871	0.075821	0.042002	0.033746	0.034051	**0.082235**
	All	Random	R_A	0.050496	0.023355	0.032433	0.009737	0.049260	0.100098	0.052479	0.048423	0.031220	0.103891
		Pareto	P_A	0.000370	0.000870	0.007895	0.000223	0.026810	0.079550	0.038438	0.039053	0.036940	**0.080524**
		HC	H_A	0.000464	0.001123	0.006643	0.000181	0.033843	0.066836	0.040360	0.037745	0.035836	**0.072906**
		First	F_KM	**0. 000249**	0.000845	0.007064	0.000455	0.017047	0.062292	0.038016	**0.028172**	**0.021480**	**0.078910**
		Last	L_KM	**0. 000264**	0.000815	**0. 002635**	**0.000006**	0.021648	0.064646	0.043254	0.035481	**0.020752**	**0.077918**
	K-Middle	Random	R_KM	0.067028	0.034595	0.037654	0.010079	0.040607	0.088794	0.048227	0.040580	**0. 022628**	**0.079505**
		Pareto	P_KM	**0. 000289**	0.000786	0.004681	0.000099	0.022649	0.068433	0.037554	0.032848	**0.025542**	**0.082684**
		HC	H_KM	**0. 000296**	**0. 000739**	**0. 002824**	**0. 000201**	0.025232	0.067418	0.045563	0.035454	0.034019	**0.079289**
RM	Random Mutation		RM	**0. 000260**	**0. 000739**	**0.002330**	**0. 000077**	**0. 011844**	**0.041814**	**0.028186**	0.035835	0.041017	0.089703
CO	Crossover Operator		CO	0.005152	0.027353	0.037131	0.044826	0.062030	0.116553	0.054050	0.051028	0.056559	0.116051

Table 5
Comparison of 20 multi-objective path relinking algorithms with hypervolume-based multi-objective local search algorithm (random mutations and crossover operator) on the instances from 50_15_01 to 100_20_01. Each value in the table represents the average hypervolume difference. The first column represents the initialization mechanism. The second and third columns provide the information of solution selection and path generation. The fourth column gives a short name for each algorithm.

Initial	Selection	Path	Name	Instance									
				50_15_01	50_20_01	70_5_01	70_10_01	70_15_01	70_20_01	100_5_01	100_10_01	100_15_01	100_20_01
		First	F_B	0.108490	0.148199	0.133240	**0. 123661**	0.143313	0.153083	0.306152	0.098688	0.189059	0.214739
		Last	L_B	0.103764	0.126037	0.121751	**0. 123262**	0.142302	0.141322	0.297986	0.100784	0.182185	0.211679
	Best	Random	R_B	0.173639	0.176523	0.191452	0.177933	0.174514	0.183869	0.359023	0.121682	0.205879	0.220908
		Pareto	P_B	0.100234	0.133196	**0. 089262**	**0. 118558**	**0. 128353**	0.140264	0.299998	0.089774	0.173475	0.219688
		HC	H_B	0.104778	0.135260	0.116909	**0. 124427**	0.167430	0.144120	0.360918	0.091359	0.190744	0.205566
		First	F_M	0.139318	0.155573	0.115563	0.152522	0.164047	0.165059	0.301088	0.111690	0.193036	0.214930
		Last	L_M	0.142790	0.140565	0.105597	0.151303	0.177946	0.165122	0.300112	0.118051	0.196224	0.211885
	Middle	Random	R_M	0.156972	0.146388	0.152058	0.157369	0.164179	0.147617	0.236139	0.104086	0.175943	0.187275
		Pareto	P_M	0.154364	0.170573	0.101882	0.175553	0.188029	0.189684	0.343840	0.127498	0.213597	0.225025
		HC	H_M	0.147029	0.162535	0.129345	0.172777	0.177395	0.174432	0.327491	0.119376	0.215739	0.222325
MOPR		First	F_A	0.097741	0.108804	0.113636	**0. 128820**	**0. 135507**	**0. 116093**	0.233584	**0. 080108**	0.151594	0.171180
		Last	L_A	**0. 090233**	**0. 096020**	0.152684	0.127642	0.149083	**0. 106867**	0.223070	0.086730	0.173461	**0.149535**
	All	Random	R_A	0.131563	0.129671	0.110650	0.131195	0.149831	0.139377	0.199309	0.093883	0.187296	0.205930
		Pareto	P_A	**0. 097386**	**0.091604**	**0. 096338**	**0. 128137**	0.147742	**0. 103385**	0.233551	0.083745	0.154365	**0.142932**
		HC	H_A	0.099880	0.098633	0.100496	**0. 111048**	0.152797	**0. 108392**	0.227072	**0. 086715**	0.158270	0.164329
		First	F_KM	**0. 091511**	**0. 101883**	**0.073433**	**0. 123691**	0.142028	**0. 103889**	**0. 190874**	0.087034	**0.127762**	**0.147002**
		Last	L_KM	**0.085316**	**0. 102233**	0.106567	**0. 120309**	0.145111	**0. 110403**	0.203629	0.091389	0.146833	**0.139944**
	K-Middle	Random	R_KM	**0. 091552**	**0. 093540**	**0. 096111**	**0. 119054**	**0. 134607**	**0.102067**	**0.157834**	**0.071063**	**0. 128876**	**0.131843**
		Pareto	P_KM	**0. 089902**	**0. 093254**	0.112003	**0. 120429**	**0.125050**	**0. 104788**	**0. 190107**	0.083151	**0.135515**	**0.149271**
		HC	H_KM	0.096104	**0. 097193**	**0. 090618**	**0.110738**	0.139718	**0. 109993**	**0. 182717**	**0. 083128**	0.143035	**0.156725**
RM	Random Mutation		RM	0.114880	0.117150	**0. 084047**	0.146445	0.156965	0.135491	**0. 169815**	**0. 080287**	0.163312	**0. 137246**
CO	Crossover operator		CO	0.131505	0.141695	0.146741	0.172327	0.178769	0.137697	**0.175162**	0.086577	0.174849	0.180406

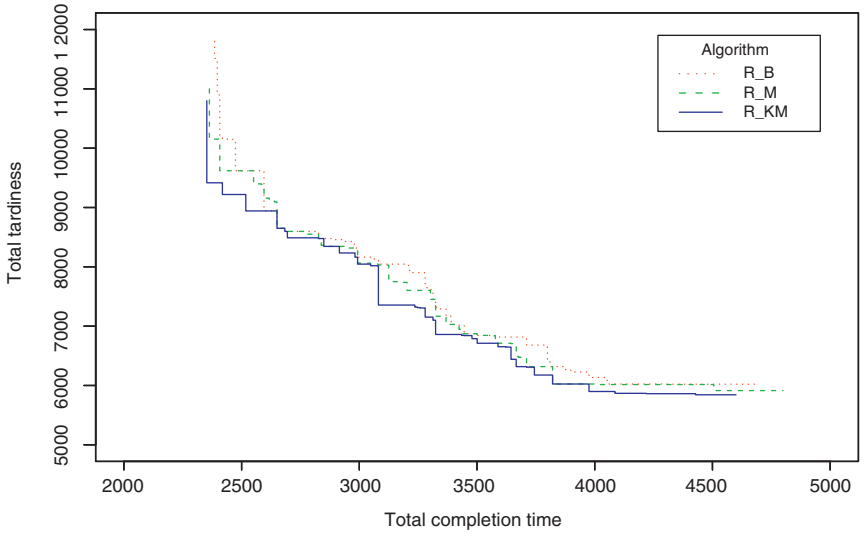

Fig. 17. Empirical attainment functions obtained by the algorithms RM, R_KM and R_B on the instance 20_20_01. The lines correspond to the limit of the objective space which are attained by at least 95% of the runs.

longer, which increases the possibility that we can explore the new high quality areas in the search space, Then, we obtain the better results, especially for the *K*-Middle version of HPR.

For the *K*-Middle version of HPR, the feature of the solutions in the generated path is usually different from the initial solution and guiding solutions, which reinforces the population diversity. That also explains why this version performs well on the large instances (from 70_5_01 to 100_20_01). Since N_{KM} is smaller than N_{All} (Section 3.2.4) in most cases, it saves a lot of time during the reinitializing process. Then, the algorithms F_KM, L_KM, R_KM, P_KM and H_KM perform more effectively than the algorithms F_A, L_A, R_A, P_A and H_A.

In Figs. 15 and 16, the empirical functions obtained on the instances 30_10_01 and 100_15_01 are presented. This figure illustrates the minimal values in the objective space, which are attained with at least 95% of the runs. The algorithms RM, R_KM and R_A are plotted with lines in different colors, where the *x*-axis indicates the total completion time and the *y*-axis indicates the total tardiness. On the instance 30_10_01 (Fig. 15), RM obtains better results, but on the instance 100_15_01 (Fig. 16), R_KM outperforms RM and R_A.

In the algorithms F_M, L_M, R_M, P_M and H_M, only one intermediate solution is generated at each step, which means these algorithms spend a little time in the reinitializing process. Only one solution helps to improve the diversity of the population *P*, which reduces the effectiveness of these variants. Besides, we select the best solutions in the algorithms F_B, L_B, R_B, P_B and H_B. However, these selected solutions are often close to the initial and the guiding solutions, it also leads to small improvements of these variants, which decreases the global effectiveness of these algorithms.

As illustrated in Figs. 17 and 18, R_KM shows its superiority against R_B and R_M on two instances 20_20_01 and 100_5_01. On the small instance (i.e., 20_20_01), the results are often close to the optimal set, then the lines are close to each other. However, it is observed that the superiority of R_KM becomes more obvious when we consider the large instances such as 100_5_01.

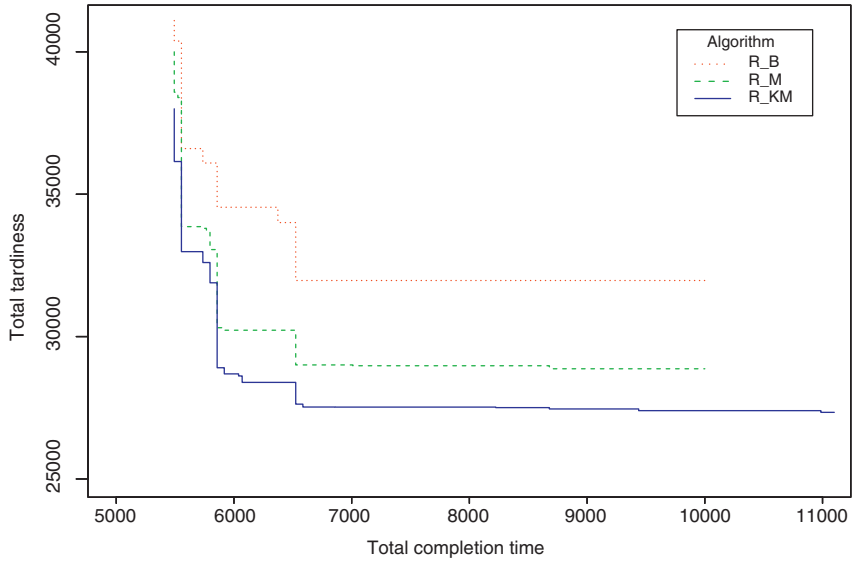

Fig. 18. Empirical attainment functions obtained by the algorithms RM, R_KM and R_B on the instance 100_5_01. The lines correspond to the limit of the objective space which are attained by at least 95% of the runs.

6. Conclusions and perspectives

In this paper, we have integrated the PR techniques into the hypervolume-based multi-objective algorithm. The proposed HPR algorithm is a combination of a local search and a PR algorithm, which are working together on two different populations. The HBMOLS algorithm aims to improve the quality of a population of solutions, and the MOPR algorithm picks up two solutions from the archive consisting of the best solutions (i.e., the non-dominated set) and selects a set of solutions from a generated path between these two solutions to initialize a new local search. Then, the solutions selected from the path are used to initialize a new population for HBMOLS. Moreover, we have focused on two important steps of MOPR: the methods of path generation and the mechanisms of solution selection for the population reinitialization. According to the combinations of these two steps, we have investigated 20 different versions of the HPR algorithm.

We have carried out experiments on the benchmark instances of the bi-objective flow shop problem. Experimental results indicate that HPR is competitive for the versions that select *K*-Middle solutions from the path as the candidate solutions to be inserted in the population, especially on the large instances. Analysis of the different versions of HPR sheds light on the ways to future research. One immediate possibility is to apply these algorithms to other multi-objective problems such as multi-objective quadratic assignment problem. In fact, we provide a general scheme of path relinking algorithm, which can be used to solve different optimization problems. Additionally, we can also consider how to select more than two solutions to generate the paths [12]. Considering the flow shop problem, we can use other distance measures and neighborhood operators for path construction. Finally, it would be interesting to study other mechanisms of solution selection for further improvements.

Acknowledgment

The work is partially supported by the “Pays de la Loire” Region (France) within the RaDaPop (2009–2013) and LigeRO (2010–2013) projects.

References

[1] H. Amin-Tahmasbi, R. Tavakkoli-Moghaddam, Solving a bi-objective flowshop scheduling problem by a multi-objective immune system and comparing with SPEA2+ and SPGA, Advances in Engineering Software 42 (10) (2011) 772–779.
[2] M. Basseur, E.K. Burke, Indicator-based multiobjective local search., in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 2007 September, pp. 3100–3107.
[3] M. Basseur, A. Liefooghe, K. Le, E. Burke, The efficiency of indicator-based local search for multi-objective combinatorial optimisation problems, Journal of Heuristics 18 (2) (2012) 263–296.
[4] M. Basseur, Rong-Qiang Zeng, Jin-Kao Hao, Hypervolume-based multi-objective local search, Neural Computing and Applications 21 (8) (2012) 1917–1929.
[5] M. Basseur, F. Seynhaeve, E.-G. Talbi, Design of multi-objective evolutionary algorithms: application to the flow-shop scheduling problem, in: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2002), vol. 2, Honolulu, USA, 2002, pp. 1151–1156.
[6] M. Basseur, F. Seynhaeve, E.-G. Talbi, Path relinking in pareto multi-objective genetic algorithms, in: Proceedings of the 3rd International Conference on Evolutionary Multi-criterion Optimization (EMO 2005), volume 3410 of Lecture Notes in Computer Science, Guanajuato, Mexico, Springer-Verlag, 2005 March, pp. 120–134.
[7] W. Bozejko, M. Wodecki, Parallel path-relinking method for the flow shop scheduling problem, in: Proceedings of International Conference on Computational Science (ICCS) Part 1, volume 5101 of Lecture Notes in Computer Science, Springer, 2008, pp. 264–273.
[8] C. Brizuela, R. Aceves, Experimental genetic operators analysis for the multi-objective permutation flowshop, in: Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), volume 2632 of Lecture Notes in Computer Science, Springer, 2003, pp. 578–592.
[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Press, Cambridge, MA, USA, 1990.
[10] W.E. Costa, M.C. Goldbarg, E.G. Goldbarg, Grasp with path-relinking adapted to total flowtime permutation flowshop schedulding problem, Department de Informatica e Matematica Aplicada Universidade Federal do Rio Grande do Norte, 2011 (technical report).
[11] J. Du, J.Y.-T. Leung, Minimizing total tardiness on one machine is NP-hard, Mathematics of Operations Research 15 (1990) 483–495.
[12] F. Glover, M. Laguna, Fundamentals of scatter search and path relinking, Control and Cybernetics 29 (1999) 653–684.
[13] F. Glover, M. Laguna, R. Marti, Scatter search, Theory and Applications of Evolutionary Computation: Recent Trends (2002) 519–529.
[14] T. James, C. Rego, F. Glover, Sequential and parallel path-relinking algorithms for the quadratic assignment problem, IEEE Intelligent Systems 20 (4) (2005) 58–65.
[15] M. Khalili, R. Tavakkoli-Moghaddam, A multi-objective electromagnetism algorithm for a bi-objective flowshop scheduling problem, Journal of Manufacturing Systems 31 (2) (2012) 232–239.
[16] J. Lemesre, C. Dhaenens, E. Talbi, An exact parallel method for a bi-objective permutation flowshop problem, European Journal of Operational Research 177 (3) (2007) 1641–1655.
[17] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics 1 (1977) 343–362.
[18] A. Liefooghe, M. Basseur, J. Humeau, L. Jourdan, E. Talbi, On optimizing a bi-objective flowshop scheduling problem in an uncertain environment, Computers and Mathematics with Applications 64 (12) (2012) 3747–3762.
[19] E. Nowicki, C. Smutnicki, A fast tabu search algorithm for the permutation flow-shop problem, European Journal of Operational Research 91 (1) (1996) 160–175.
[20] M. Pasia, X. Gandibleux, F. Doerner, F. Hartl, Local search guided by path relinking and heuristic bounds, in: Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization (EMO 2007), volume 4403 of Lecture Notes in Computer Science, Springer, 2007, pp. 501–515.
[21] R. Reeves, T. Yamada, Genetic algorithms, path relinking and the flowshop sequencing problem, Evolutionary Computation 6 (1) (1998) 45–60.
[22] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European Journal of Operational Research 177 (2007) 2033–2049.
[23] T. Schiavinotto, T. Stützle, A review of metrics on permutations for search landscape analysis, Computers and Operations Research 34 (10) (2011) 3143–3153.
[24] M. Sevaux, K. Sörensen, Permutation distance measures for memetic algorithms with population management, in: Proceedings of the 6th Metaheuristics International Conference (MIC'2005), Vienna, Austria, 2005, pp. 22–26.
[25] E. Taillard, Some efficient heuristic methods for flow-shop sequencing, European Journal of Operational Research 47 (1990) 65–74.
[26] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (1993) 278–285.
[27] E. Vallada, R. Ruiz, Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem, Omega 38 (2010) 57–67.
[28] E. Zitzler, Evolutionary algorithms for multiobjective optimization: methods and applications, ETH, Zurich, 1999 (PhD thesis).
[29] E. Zitzler, S. Kuenzli, Indicator-based selection in multiobjective search, in: Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Birmingham, UK, 2004 September, pp. 832–842.
[30] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, Evolutionary Computation 3 (1999) 257–271.