Hypervolume-Based Multi-Objective Path Relinking Algorithm

Rong-Qiang Zeng, Matthieu Basseur, and Jin-Kao Hao

LERIA, Université d'Angers, 2, Boulevard Lavoisier,
49045 Angers Cedex 01, France
{zeng,basseur,hao}@info.univ-angers.fr

Abstract. This paper presents a hypervolume-based multi-objective path relinking algorithm for approximating the Pareto optimal set of multi-objective combinatorial optimization problems. We focus on integrating path relinking techniques within a multi-objective local search as an initialization function. Then, we carry out a range of experiments on bi-objective flow shop problem and bi-objective quadratic assignment problem. Experimental results and a statistical comparison are reported in the paper. In comparison with the other algorithms, one version of our proposed algorithm is very competitive. Some directions for future research are highlighted.

Keywords: multi-objective optimization, hypervolume contribution, path relinking, local search, flow shop problem, quadratic assignment problem.

1 Introduction

Local search is an effective search strategy for both single objective optimization and multi-objective optimization. Particularly, local search requires a method to generate initial solutions. However, how to set the initialization methods still remains an open question in many cases, especially in multi-objective optimization. In this paper, we investigate path relinking [8] as an initialization method for hypervolume-based multi-objective local search (HBMOLS) [3].

The HBMOLS algorithm aims to generate a Pareto approximation set by improving an initial population. In this work, we use path relinking to construct paths and then select from each path a set of solutions to initialize a new population for HBMOLS. In order to evaluate the effectiveness of our proposed method, we show experimental results on the bi-objective flow shop problem and bi-objective quadratic assignment problem, and we compare them with the HBMOLS algorithm which initializes a new population using random mutations or crossover operator.

The remainder of this paper is organized as follows. In Section 2, we present some basic notations and definitions related to multi-objective optimization. Then, in Section 3, we briefly review the literature using the path relinking techniques to solve multi-objective optimization problems. Afterwards, in Section 4, we describe the hypervolume-based multi-objective path relinking algorithm. Section 5 reports the computational results and analyzes the behavior of the proposed algorithm. Finally, the conclusions and perspectives are given in the last section.

R.C. Purshouse et al. (Eds.): EMO 2013, LNCS 7811, pp. 185–199, 2013.

2 Multi-Objective Optimization

In this section, we recall some useful notations and definitions of multi-objective optimization. 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 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:

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. *$x \in S$ $(S \subset X)$ is said to be non-dominated if and only if there does not exist another solution $x^{'} \in S$ such that $x^{'}$ dominates x. When $S \equiv X$, x is said to be Pareto optimal.*

Definition 3. *S is said to be a non-dominated set if and only if S is composed of non-dominated solutions. When S is composed of all the Pareto optimal solutions, S is said to be a Pareto optimal set.*

In multi-objective optimization, there usually does not exist one optimal but a set of Pareto optimal solutions, which keeps the best compromise among all the objectives. Nevertheless, in most cases, it is not possible to compute the Pareto optimal set in a reasonable time. Then, we are interested in computing a non-dominated set, which is as close to the Pareto optimal set as possible. Therefore, the goal is often to identify a good Pareto approximation set.

3 Related Works

Path Relinking (PR) was initially proposed by Glover [8] as an effective search strategy, which has proved its efficiency in single objective optimization [8]. Its objective is to explore the search space by creating paths within a given set of high-quality solutions. In the following paragraphs, we focus on the studies dealing with multi-objective optimization problems.

Basseur et al. [2] propose a multi-objective approach to integrate PR techniques into an adaptive genetic algorithm, which is dedicated to obtaining a first well diversified Pareto approximation set. Based on this set, two solutions are randomly selected to generate a path. According to the distance measure defined in [2], there are many intermediate solutions which can be generated at each step of the PR procedure. Then, the authors apply a random aggregation of the objectives to determine which solution is selected from the possible 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 flow shop problem show that this PR approach is very promising and efficient.

In [13], Pasia et al. present three PR approaches for solving a bi-objective flow shop problem. By using a straightforward implementation of the ant colony system, they first generate two pools of initial solutions, where one pool contains solutions that are good with respect to the makespan and the other one 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, the authors randomly select two solutions from this non-dominated set to construct a path. Along the path, some of the solutions are submitted for improvements. 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 the local nadir points. Computational results demonstrate that their proposed approaches are competitive.

In addition, two different versions of iterated Pareto local search (IPLS) algorithms, which are path-guided IPLS (pIPLS) and a combination of IPLS and pIPLS named rIPLS, are presented in [6]. The authors propose a path-guided mutation that generates solutions on the path linking two local optimal individuals. This mutation generates individuals at a certain distance from the initial solution to the guiding solution. Then, Pareto local search is restarted from the individual generated on the path. Experiments on bi-objective quadratic assignment problem show that pIPLS and rIPLS both outperform the multi-restart Pareto local search algorithm.

4 Hypervolume-Based Multi-Objective Path Relinking Algorithm

This section describes the hypervolume-based multi-objective path relinking algorithm, which is a combination of the Hypervolume-Based Multi-Objective Local Search algorithm (HBMOLS) and the Multi-Objective Path Relinking algorithm (MOPR). The outline of the proposed algorithm is illustrated in Algorithm 1 and depicted in Fig. 1.

In this algorithm, all the solutions in an initial population are randomly generated. Then, each solution in the population is optimized by the HBMOLS algorithm [3], which is based on the Hypervolume Contribution Selection illustrated in Algorithm 2. The HBMOLS algorithm achieves the fitness assignment by using the hypervolume contribution indicator $HC(x, P)$ defined in [3]. Afterwards, we randomly choose two solutions (an initial solution and a guiding solution) from the Pareto approximation set generated by HBMOLS, and we define a distance between these two solutions to construct a path. At each step, we generate only one new solution and make sure the distance between the new solution and the guiding solution decreases by 1.

After the path generation, a subset of solutions in the path are selected and used to initialize a new population P for HBMOLS. These solutions are potentially inserted into P, according to their corresponding hypervolume contribution. Actually, we propose four mechanisms to select a set of solutions from the generated path. These mechanisms are illustrated in Fig. 2 and described in detail below.

All: All the solutions in the path are selected to be inserted into the population P (solutions represented both in circle and in square in Fig. 2).

Algorithm 1. Hypervolume-Based Multi-Objective Path Relinking Algorithm

Input: N (Population size)
Output: A (Pareto approximation set)
Initialization: $P \leftarrow N$ randomly generated solutions
$A \leftarrow$ Non dominated solutions of P
while Running time is not reached **do**
Local Search (HBMOLS):
1) Fitness Assignment: Calculate a fitness value for each $x \in P$, i.e., $Fit(x) = HC(x, P)$
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
3) $A \leftarrow$ Non dominated solutions of $A \bigcup P$. If A does change, back to step 2
Path Relinking (MOPR):
1) $P^{'} \leftarrow N$ randomly generated solutions
2) randomly choose an initial solution x_i and a 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 a path linking x_i to x_j
5) select n_{pr} solutions: $T^{'} = \{y_1, y_2, \cdots, y_{n_{pr}}\}$ from the set T
6) **for** $i \leftarrow 1, \ldots, n_{pr}$ **do**
Hypervolume Contribution Selection $(P^{'}, y_i)$
end for
end while
Return A

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. 2, the solutions represented in square are selected, since they belong to 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. These solutions could not be very useful, since HBMOLS will search the explored areas alike. One way to avoid this problem is to select a single solution, which is located in the middle of the path (solution represented in black circle in Fig. 2). In fact, this mechanism can be seen as a kind of crossover operator.

K**-Middle:** Here, we also aim to avoid the problem of proximity of intermediate solutions to the initial solution and the guiding solution. 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 generated path. We define this number by 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 not bigger than $\sqrt{N_{All}}$ (solutions located in the dashed circle in Fig. 2).

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

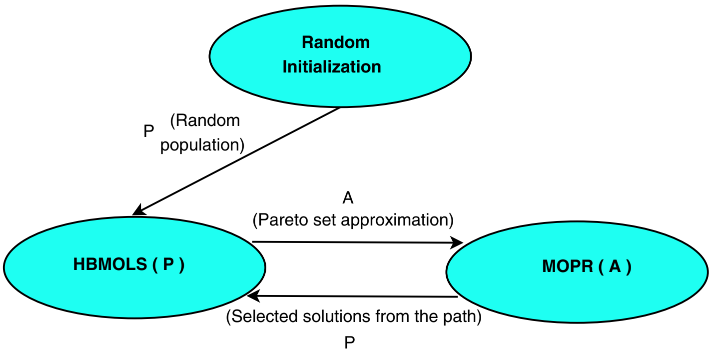

Fig. 1. A random population is initialized and provided as an entry to to HBMOLS, which generates a Pareto approximation set by improving the initial population. Then, MOPR generates a path between two solutions belonging to the Pareto approximation set provided by HBMOLS. A subset of solutions in the path is selected to initiate a new HBMOLS execution.

5 Computational Results

In order to evaluate the efficiency of our proposed algorithms, we carry out experiments on the bi-objective flow shop problem and bi-objective quadratic assignment problem. We compare four versions of hypervolume-based multi-objective path relinking algorithm (named PR_A, PR_B, PR_M and PR_KM) with two versions of HBMOLS (named RM and CO), which use random mutation and crossover operator as the initialization functions [1]. All the algorithms are programmed in C and compiled using Dev-C++ on a PC running Windows XP with Pentium 2.61 GHz CPU and 2 GB RAM.

5.1 Performance Assessment Protocol

We evaluate the effectiveness of multi-objective optimization algorithms by using a test procedure that has been undertaken with the performance assessment package provided by Zitzler et al.[1]

The quality assessment protocol works as follows: we first create a set of 20 runs with different initial populations for each algorithm and each benchmark instance. Afterwards, we calculate the set PO^* in order to determine the quality of k different sets

[1] `http://www.tik.ee.ethz.ch/pisa/assessment.html`

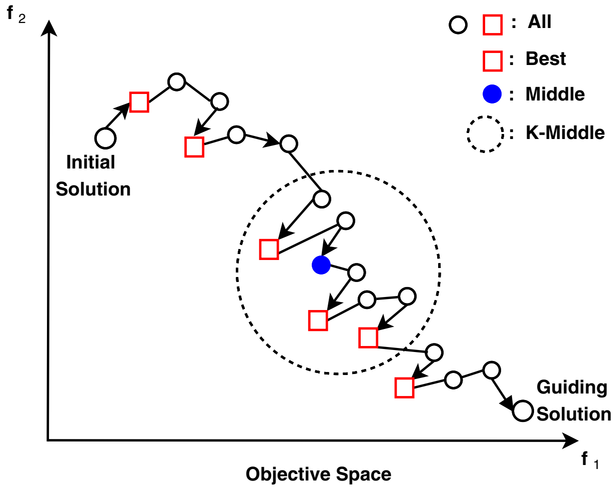

Fig. 2. The mechanisms of subset selection

$A_0 \dots A_{k-1}$ of non-dominated solutions (The set PO^* is generated by removing the dominated solutions from the union of k different sets, more details can be found in [19]). Furthermore, we define a reference point $z = [w_1, w_2]$, where w_1 and w_2 represent the worst values for each objective function in $A_0 \cup \cdots \cup A_{k-1}$. Then, the evaluation of a set A_i of solutions can be determined by finding the hypervolume difference between A_i and PO^* [19], which has to be as close to zero as possible.

For each algorithm, we compute 20 hypervolume differences corresponding to 20 runs, and perform the Mann-Whitney statistical test on the sets of hypervolume difference. In our experiments, we say that an algorithm $\mathcal{A}$ outperforms an algorithm $\mathcal{B}$ if the Mann-Whitney test provides a confidence level greater than 95%. The computational results are summarized in Tables 2 and 4 respectively. In these two tables, each line contains at least a value **in grey** for each instance, which corresponds to the best average hypervolume difference obtained by the corresponding algorithm. 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%).

5.2 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. In this paper, we focus on optimizing two objectives: total completion time and total tardiness.

5.2.1 Bi-objective Flow Shop Problem

Generally, the FSP deals with n jobs $\{J_1, J_2, ..., J_n\}$ and m machines $\{M_1, M_2, ..., M_m\}$, 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.

Specifically, each job J_i is composed of m consecutive tasks $\{t_{i1}, t_{i2}, ..., t_{im}\}$, where t_{ij} represents the j^{th} task of the job J_i requiring the machine m_j. Each task t_{ij} is associated with a processing time p_{ij}, which is scheduled at the time s_{ij} and should be achieved before the due date d_j. Actually, we aim to minimize two objective functions: total completion time C_{max} and total tardiness T, which are formally defined as follows:

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

$$f_2 = T = \sum_{i=1}^{n} [\max(0, s_{im} + p_{im} - d_i)] \tag{2}$$

Both of them have been proven to be NP-hard [9,7]. In addition, all the FSP instances used in this paper are taken from Taillard benchmark instances and extended into bi-objective case [17][2].

5.2.2 Path Generation

A candidate solution to FSP can be encoded as a permutation $\mathcal{P}$ composed of $\{0, \ldots, n-1\}$ values, such that $\mathcal{P}(i)$ denotes the job to be executed at the i^{th} position. As proved in [15], the insertion operator, which inserts a selected job to a designated position, is more effective than other operators in solving FSP. Moreover, the authors in [4] show the insertion operator is also very efficient in solving multi-objective FSPs.

Therefore, we decide to define our distance measure directly related to the insertion operator. This property allows us to to compute the minimum number of moves, which have to be applied on an initial solution to reach a guiding solution. As suggested in [2], we use the Longest Common Subsequence (LCS) between two solutions as a distance measure for path generation. The LCS can be calculated in $O(n^2)$ by a dynamic programming algorithm, which is similar to the well known Needleman-Wunsch algorithm [5,14]. Then, the distance between two solutions is defined as the length of permutation minus the length of LCS.

After the distance computation, we generate a path in a random way. In this method, we randomly select a candidate job, and insert this job into a randomly selected position. In fact, this method consists of four main steps:

Step 1: We randomly select a candidate job from an initial solution. For example, in Fig. 3, the longest common subsequence between an initial solution and a guiding solution is colored in black, the remaining jobs are candidate jobs. In this example, the candidate job 15 is randomly selected.

[2] Benchmarks available at
`http://www.lifl.fr/ liefooga/benchmarks/index.html`

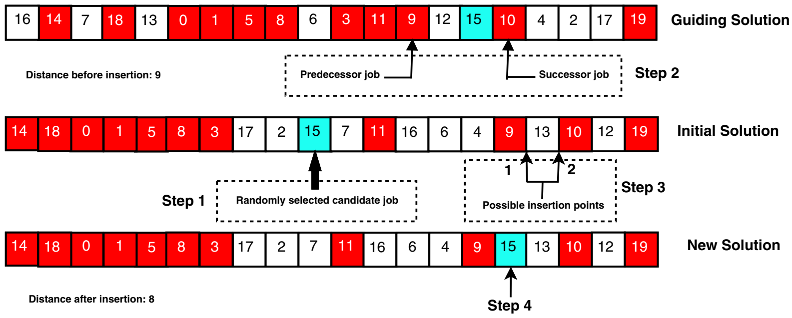

Fig. 3. Path generation for flow shop problem

Step 2: We find the position of the selected candidate job in the LCS of the guiding solution. In Fig. 3, the candidate job 15 is located between two jobs 9 and 10.

Step 3: We find the insertion position for the selected candidate job in the LCS of the initial solution. As shown in Fig. 3, there are two possible insertion positions for the job 15: (9 13) and (13 10).

Step 4: We insert the selected candidate job into a randomly selected insertion position to generate a new solution in the path. As illustrated in Fig. 3, we insert the job 15 into the randomly selected insertion position (9 13) to obtain a new solution. We continue the process in this manner until the distance between the new solution and the guiding solution equals to 0.

5.2.3 Parameters Settings

The proposed algorithms require to set a few parameters, we mainly discuss two important ones: running time and population size.

Running time: The running time T is a key parameter in the experiments. We define the time T for each instance by Equation 3, in which N_{Job} and N_{Mac} represent the number of jobs and the number of machines of one instance, N_{Obj} represents the number of objectives (see Table 1).

$$T = \frac{{N_{Job}}^2 \times N_{Mac} \times N_{Obj}}{100} \ sec \tag{3}$$

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

Population size: According to the results obtained in [1], the experiments realized previously on the IBMOLS algorithm showed that the best results are achieved with a small population size N. We set this size from 10 to 40 individuals by Equation 4, relative to the size of tested instance (see Table 1).

$$|N| = \begin{cases} 10 & : \quad 0 < |N_{Job} \times N_{Mac}| < 500 \\ 20 & : \quad 500 \leq |N_{Job} \times N_{Mac}| < 1000 \\ 30 & : \quad 1000 \leq |N_{Job} \times N_{Mac}| < 2000 \\ 40 & : \quad 2000 \leq |N_{Job} \times N_{Mac}| < 3000 \end{cases} \tag{4}$$

Table 1. Parameter values used for bi-objective FSP instances (i_j_k represents the k^{th} bi-objective FSP 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”

5.2.4 Experimental Results

The computational results are summarized in Table 2. In this table, we observe that RM has a good performance on the first eight instances from 20_5_01 to 30_20_01. It obtains the best average hypervolume differences on these instances. On the other hand, PR_KM outperforms the other algorithms on the remaining instances from 50_5_01 to 100_20_01, where almost all the best results are obtained by this algorithm. Additionally, CO is less effective in comparison with RM and PR_KM.

From Table 2, we can see the path relinking techniques have a limited contribution on the small instances from 20_5_01 to 30_20_01. We suppose that, when the instance size 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 initialize a new population. In this case, it is more useful to perform 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 is longer, which means we have more possibilities to explore new high quality areas in the search space. Therefore, PR_KM has a good performance on the large instances from 50_5_01 to 100_20_01.

Table 2. Comparison of four versions of hypervolume-based multi-objective path relinking algorithm (PR_A, PR_B, PR_M and PR_KM) with two versions of HBMOLS (RM and CO) on 20 bi-objective FSP instances from 20_5_01 to 100_20_01. Each value in the table represents an average hypervolume difference.

Instance	Algorithm					
	PR_A	PR_B	PR_M	PR_KM	RM	CO
20_05_01_ta001	0.050496	0.076627	0.093801	0.067028	**0.000260**	0.005152
20_10_01_ta011	0.023355	0.055498	0.048349	0.034595	**0.000739**	0.027353
20_15_01	0.032433	0.073174	0.070448	0.037654	**0.002330**	0.037131
20_20_01_ta021	0.009737	0.034508	0.024761	0.010079	**0.000077**	0.044826
30_05_01	0.049260	0.081154	0.099705	0.040607	**0.011844**	0.062030
30_10_01	0.100098	0.200979	0.176367	0.088794	**0.041814**	0.116553
30_15_01	0.052479	0.096203	0.105293	0.048227	**0.028186**	0.054050
30_20_01	0.048423	0.064844	0.071167	***0.040580***	**0.035835**	0.051028
50_05_01_ta031	0.031220	0.083466	0.090345	**0.022628**	0.041017	0.056559
50_10_01_ta041	0.103891	0.149919	0.132192	**0.079505**	0.089703	0.116051
50_15_01	0.131563	0.173639	0.156972	**0.091552**	0.114880	0.131505
50_20_01_ta051	0.129671	0.176523	0.146388	**0.093540**	0.117150	0.141695
70_05_01	0.110650	0.191452	0.152058	***0.096111***	**0.084047**	0.146741
70_10_01	0.131195	0.177933	0.157369	**0.119054**	0.146445	0.172327
70_15_01	0.149831	0.174514	0.164179	**0.134607**	0.156965	0.178769
70_20_01	0.139377	0.183869	0.147617	**0.102067**	0.135491	0.137697
100_05_01_ta061	0.199309	0.359023	0.236139	**0.157834**	***0.169815***	***0.175162***
100_10_01_ta071	0.093883	0.121682	0.104086	**0.071063**	***0.080287***	0.086577
100_15_01	0.187296	0.205879	0.175943	**0.128876**	0.163312	0.174849
100_20_01_ta081	0.205930	0.220908	0.187275	**0.131843**	***0.137246***	0.180406

Compared with other versions of hypervolume-based multi-objective path relinking algorithms, the advantages of PR_KM are very clear. As N_{KM} is smaller than N_{All}, in most cases, PR_KM saves a lot of time during the initializing process, then it performs more effectively than PR_A, especially on the large instances. Considering PR_B, we select a set of non-dominated solutions from the path. However, these solutions are often close to the initial solution and the guiding solution. The similar search areas have little contribution in initializing a new population, which decreases the global effectiveness of PR_B. For PR_M, only one intermediate solution is selected from the path at each step, which means this algorithm spends a little time in the initializing process. Then, it is not very helpful to reinforce the population's diversity. For this reason, the effectiveness of PR_M is affected.

5.3 Application to Bi-objective Quadratic Assignment Problem

The quadratic assignment problem (QAP) is a classical combinatorial optimization problem both in theory and in practice. As one of the most difficult problems in the NP-hard class, it models many real-life problems in many areas such as the facility location, parallel and distribute computing, and combinatorial data analysis [11]. In our case, we concentrate on bi-objective quadratic assignment problem.

5.3.1 Bi-objective Quadratic Assignment Problem

The quadratic assignment problem can be described as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities [12]. Given n facilities and n locations, three $n \times n$ matrices D, F_1 and F_2, where d_{ij} is the distance between location i and j, and f^1_{rs} and f^2_{rs} are two flows between two facilities r and s. The goal is to minimize the sum of the product between flows and distances. The objective of the QAP can then be formulated as follows:

$$\min_{\phi \in \Phi} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} f^k_{\phi_i \phi_j}, \; k \in \{1, 2\} \tag{5}$$

where Φ is the set of all permutations of $\{1, \ldots, n\}$, and ϕ_i gives the location of item i in a solution $\phi \in \Phi$.

In this paper, all the tested instances of QAP are provided by R. E. Burkard et al.[3] In our case, a bi-objective QAP instance is generated by keeping the distance matrix of the first instance and using two different flow matrices. Moreover, we denote a bi-objective instance as N_i_ab (N represents the name of instance such as "esc") with a matrix of size i respectively. For example, esc_32_ab denotes a bi-objective instance named "esc", which is generated by two single-objective instances esc_32_a and esc_32_b.

5.3.2 Path Generation

A candidate solution to QAP can be encoded as a permutation $\mathcal{P}$ composed of $\{1, \ldots, n\}$ values, such that $\mathcal{P}(i)$ denotes the facility to be assigned at the i^{th} location. As proved in [16], the swap operator, which exchanges two facilities in a permutation, is very effective for solving QAP. Then, we define the distance between two solutions directly related to the swap operator.

For QAP, we use the permutation distance and the cycle distance [18,14] as the distance measure. Actually, the distance between two solutions is defined as the permutation distance minus the cycle distance. Afterwards, we construct a path by randomly selecting an element from one cycle in a permutation and applying the swap operator to this element to obtain a new solution.

An example of path generation for QAP is illustrated in Fig. 4. In this example, there is one integer element (11) located at the same position in an initial solution and a guiding solution, then the permutation distance is 10. On the other hand, there are three cycles ($\{3, 1, 2, 7, 8\}$, $\{4, 5, 6\}$ and $\{10, 9\}$) between these two permutations, so the cycle distance is 3. Therefore, the distance between the initial solution and the guiding solution is equal to 7.

Furthermore, there are 7 steps starting from the initial solution P_x to the guiding solution P_y, which allows us to generate 6 solutions on the path. For instance, we first randomly select a facility 2 from one cycle $\{3, 1, 2, 7, 8\}$ in P_x, and we can observe the facility 2 is located at the second position in P_y. Then, we apply the swap operator to

[3] Benchmarks available at `http://www.seas.upenn.edu/qaplib/inst.html`

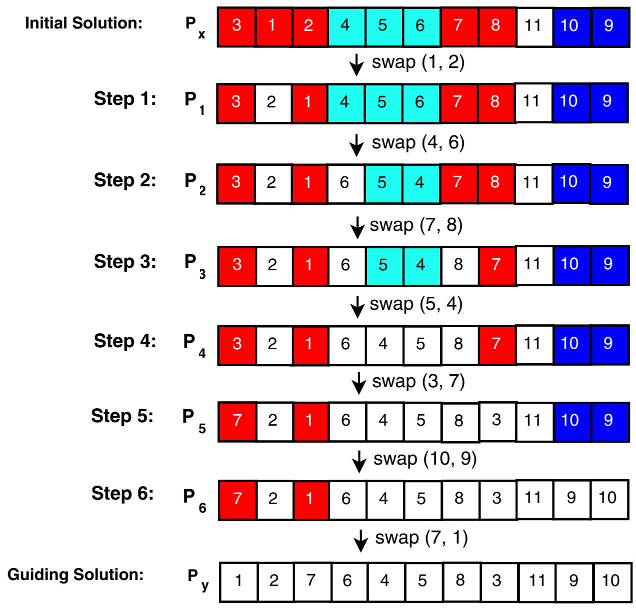

Fig. 4. Path generation for quadratic assignment problem

two facilities 1 and 2 in P_x in order to generate a new solution. We continue this process until the distance between the new solution P_i and the guiding solution P_y is equal to 0.

5.3.3 Parameters Settings

Similar to the parameter settings in FSP, we consider two important parameters: running time and population size.

– **Running time:** We define the running time T for each instance by Equation 6, in which N_{Dis}, N_{Flow} and N_{Obj} represent respectively the size of the distance matrix, the size of the flow matrix and the number of objectives in an instance (see Table 3).

$$T = N_{Dis} \times N_{Flow} \times N_{Obj} \ sec \tag{6}$$

– **Population size:** Here, we set this size from 10 to 30 individuals according to Equation 7, relatively to the size of the tested instance (see table 3).

$$|N| = \begin{cases} 10 & : \quad 0 < |N_{Dis} \times N_{Flow}| < 500 \\ 20 & : \quad 500 \leq |N_{Dis} \times N_{Flow}| < 1000 \\ 30 & : \quad 1000 \leq |N_{Dis} \times N_{Flow}| < 2000 \\ 40 & : \quad 2000 \leq |N_{Dis} \times N_{Flow}| < 3000 \end{cases} \tag{7}$$

Table 3. The instances of bi-objective quadratic assignment problem (Parameters: population size N, running time T)

Inst 1	**Inst 2**	**Dim**	**N**	**T**
chr_12_a	chr_12_b	12×12	10	4'48"
chr_15_a	chr_15_b	15×15	10	7'30"
chr_20_a	chr_20_b	20×20	10	13'20"
esc_16_a	esc_16_b	16×16	10	8'32"
esc_32_a	esc_32_b	32×32	30	16'20"
Lipa_30_a	Lipa_30_b	30×30	20	15'
Ste_36_a	Ste_36_b	36×36	30	21'36"
tai_40_a	tai_40_b	40×40	30	26'40"
tai_50_a	tai_50_b	50×50	40	41'40"

5.3.4 Experimental Results

The computational results for the bi-objective QAP are presented in Table 4. From this table, we can see RM has a good performance almost on all the instances. Particularly, it obtains the best average hypervolume differences on five instances. Moreover, PR_KM also obtains very competitive results on all the instances, especially on the large instances, such as Lipa_30_ab, tai_40_ab and tai_50_ab. However, CO is statistically outperformed by RM and PR_KM on most of the instances.

Table 4. Comparison of four versions of hypervolume-based multi-objective path relinking algorithm (PR_A, PR_B, PR_M and PR_KM) with two versions of HBMOLS (RM and CO) on 9 bi-objective QAP instances. Each value in the table represents an average hypervolume difference.

Instance	Algorithm					
	PR_A	PR_B	PR_M	PR_KM	RM	CO
chr_12_ab	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	0.013407
chr_15_ab	0.002988	0.010994	**0.000000**	0.002271	**0.000000**	0.026494
chr_20_ab	0.014042	0.025258	0.004827	0.005560	**0.001899**	0.017890
esc_16_ab	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**
esc_32_ab	0.006312	0.008839	0.002594	0.003003	**0.002433**	0.007948
Lipa_30_ab	0.001956	0.002159	***0.001369***	**0.001347**	0.001433	0.003047
Ste_36_ab	0.304087	0.356747	0.669592	0.364021	***0.215314***	**0.203776**
tai_40_ab	0.037541	0.041880	0.031969	**0.027092**	0.046076	0.080534
tai_50_ab	0.038647	0.030516	0.040565	**0.027077**	0.048410	0.046201

According to the experimental results in table 4, RM has a better performance than PR_KM on the first four instances. Since these instances are small and relatively easy to solve, the PR_KM and RM algorithms achieve the best results on three instances (chr_12_ab, chr_15_ab and esc_16_ab), where the average hypervolume differences are equal to 0. Furthermore, on these small instances, it is not easy for PR_KM to construct a long path to find enough diversified solutions for initializing a new population. Then it is better to perform random moves in the search space or to select only one solution from the generated path as done in PR_M. When the size of instance becomes larger, we can construct a longer path and select more useful solutions from the path, which means we have more chances to explore high quality areas in the objective space. Therefore, PR_KM obtains the best value on the large instances such as tai_50_ab and a competitive value on the instance esc_32_ab. However, the instance Ste_36_ab is an exception, CO obtains the best value on this instance. In fact, only several non-dominated solutions are found in the population. We suppose that the search procedure is often trapped in some local optimums, then using crossover operator is a better way to be out of these traps.

6 Conclusions and Perspectives

In this paper, we present a hypervolume-based multi-objective path relinking algorithm, which is applied to the bi-objective flow shop problem and bi-objective quadratic assignment problem. This algorithm integrates the path relinking techniques into hypervol -umebased multi-objective local search as an initialization function, in order to find a Pareto approximation set. Actually, we provide a general scheme of path relinking algorithm, which can be used to deal with other multi-objective optimization problems.

Experimental results indicate one version of our proposed algorithms is very competitive in comparison with other algorithms. The performance analysis gives us a few directions for future research. The first possibility is to generate more intermediate solutions at each step, then one can construct several different paths simultaneously. Especially, for each path, it could give birth to another path in reverse direction. Second, it is worth proposing other mechanisms of subset selection. The new mechanisms could have the potential to obtain a better Pareto approximation set.

On the other hand, it should be very interesting to integrate MOPR into other metaheuristics such as tabu search, in order to evaluate its overall effectiveness. The cooperation of MOPR with exact methods can be also a promising search area. For instance, MOPR could be used to link Pareto optimal solutions found by an exact approach. Several approaches between MOPR and exact approaches could be defined, as those described in the taxonomy of Jourdan et al. [10].

Acknowledgment. The work is partially supported by the "Pays de la Loire" Region (France) within the RaDaPop (2009-2013) and LigeRO (2010-2013) projects. We thank the reviewers of the paper for their helpful comments.

References

1. Basseur, M., Liefooghe, A., Le, K., Burke, E.: The efficiency of indicator-based local search for multi-objective combinatorial optimisation problems. Journal of Heuristics 18(2), 263–296 (2012)

2. Basseur, M., Seynhaeve, F., Talbi, E.-G.: Path Relinking in Pareto Multi-objective Genetic Algorithms. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 120–134. Springer, Heidelberg (2005)
3. Basseur, M., Zeng, R.Q., Hao, J.K.: Hypervolume-based multi-objective local search. Neural Computing and Applications 21(8), 1917–1929 (2012)
4. Brizuela, C., Aceves, R.: Experimental Genetic Operators Analysis for the Multi-objective Permutation Flowshop. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 578–592. Springer, Heidelberg (2003)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. The MIT Press, Cambrige (1990)
6. Drugan, M.M., Thierens, D.: Path-Guided Mutation for Stochastic Pareto Local Search Algorithms. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 485–495. Springer, Heidelberg (2010)
7. Du, J., Leung, J.Y.-T.: Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research 15, 483–495 (1990)
8. Glover, F., Laguna, M.: Fundamentals of scatter search and path relinking. Control and Cybernetics 29, 653–684 (1999)
9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5, 287–326 (1979)
10. Jourdan, L., Basseur, M., Talbi, E.: Hybridizing exact methods and metaheuristics: A taxonomy. European Journal of Operational Research 199(3), 620–629 (2009)
11. Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Querido, P., Querido, T.: A survey for the quadratic assignment problem. European Journal of Operational Reasearch 176, 657–690 (2007)
12. Pardalos, P., Rendl, F., Wolkowicz, H.: The quadratic assignment problem: A survey and recent developments. In: Proceedings of the DIMACS Workshop on Quadratic Assignment Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16, pp. 1–42 (1994)
13. Pasia, J.M., Gandibleux, X., Doerner, K.F., Hartl, R.F.: Local Search Guided by Path Relinking and Heuristic Bounds. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 501–515. Springer, Heidelberg (2007)
14. Schiavinotto, T., Stützle, T.: A review of metrics on permutations for search landscape analysis. Computers and Operations Research 34(10), 3143–3153 (2011)
15. Taillard, E.: Some efficient heuristic methods for flow-shop sequencing. European Journal of Operational Research 47, 65–74 (1990)
16. Taillard, E.: Robust taboo search for the quadratic assignment problem. Parallel Computing 17, 443–455 (1991)
17. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285 (1993)
18. Thierens, D.: Exploration and Exploitation Bias of Crossover and Path Relinking for Permutation Problems. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN IX. LNCS, vol. 4193, pp. 1028–1037. Springer, Heidelberg (2006)
19. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. Evolutionary Computation 3, 257–271 (1999)