Generic Hybridization of MOEA/D with Learning for Permutation Flow Shop Scheduling Problem

S. Pericleous, A. Konstantinidis, A. Achilleos
Department of Computer Science and Engineering,
Frederick University, Nicosia, Cyprus
{com.ps, com.ca, com.aa}@frederick.ac.cy

G. A. Papadopoulos
Department of Computer Science,
University of Cyprus, Nicosia, Cyprus
george@cs.ucy.ac.cy

Abstract—**In this paper, we study the effect of *Meta-Lamarckian learning* on the performance of a generic hybrid *Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D)* to solve a well-known combinatorial *Multi-Objective Optimization (MOO)* problem. We study the hybridization of MOEA/D with a pool of six *general-purpose heuristics* so as to locally optimize the solutions during the evolution. We initially consider the six individualistic hybrid MOEA/D's, in which at every step of the evolution the same local search heuristic from the generic pool is applied. MOEA/D is then enriched with a learning strategy that, based on the problem's properties and objective functions, adaptively selects at each step of the evolution and for each problem neighbourhood the best performing local search heuristic from the generic pool of heuristics. The proposed method is evaluated on various test instances of a multi-objective *Permutation Flow Shop Scheduling Problem (MO-PFFSP)*: given a set of jobs and a series of machines, the corresponding processing time of each job on every machine and the due dates of each job, determine a processing order of the jobs on each machine, so as to simultaneously minimize the makespan (total completion time), and the maximum job tardiness. The results of our experimental studies suggest that the proposed method successfully learns the behaviour of individual local search heuristics during the evolution outperforming in terms of both convergence and diversity the conventional MOEA/D and the individualistic hybrid MOEA/D's. The proposed method does not utilize any problem-specific heuristics, and as a result, is readily applicable to other combinatorial MOO problems.**

Index Terms—**multi-objective optimization, evolutionary algorithms, local search, decomposition, meta-lamarckian learning, permutation flow shop scheduling problem**

I. Introduction and Related Work

Flowshop Scheduling Problem is one of the most well-studied class of scheduling problems in the literature with significant applications in real-life manufacturing systems [1]. The *Permutation Flow Shop Scheduling Problem* (**PFSSP**) is a simplified version in which given m machines, a series of n jobs has to be processed sequentially, in the same order, from the first machine to the last so as to optimize some desired objectives. Sequence changes are not allowed, so once the sequence of jobs is scheduled on the first machine, this sequence remains unchanged on the other machines. Note that the possible number of such sequences is $n!$. After completion on one machine a job joins the queue at the next machine, all queues are assumed to operate under the FIFO discipline. Each machine can process at most one job at any given time, and it can not be interrupted. Machines never breakdown and are available throughout the scheduling period. Each job is available at time zero, and can be processed by at most one machine at any given time. The set-up times of the jobs on machines are sequence independent and are included in processing times. An example of a solution of PFSSP with 8 jobs and 3 machines is shown in Figure 1.

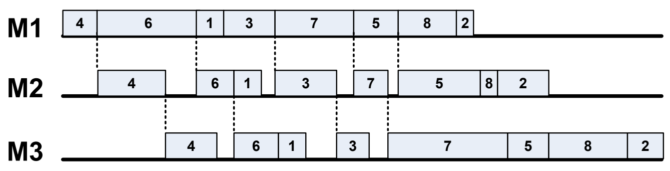

Fig. 1. Example of a solution of a PFSSP with $n = 8$ jobs and $m = 3$ machines with encoding representation $(4, 6, 1, 3, 7, 5, 8, 2)$.

Because of the conflicting nature of the objectives considered in this paper the problem will be treated within the context of Multi-Objective Optimization (MOO).

A *Multi-objective Optimization Problem* (**MOP**) can be mathematically formulated as

$$\min \ F(X) = (f_1(X), \dots, f_k(X)), \ \text{subject to} \ X \in \Omega, \quad (1)$$

where Ω is the decision space and $X \in \Omega$ is a decision vector. $F(X)$ consists of k objective functions, and $\Re^k$ is the objective space. Improving on one objective may lead to deterioration of another, thus, no single solution exists that can optimize all objectives simultaneously. The best trade-off solutions, called the set of Pareto optimal (or non-dominated) solutions, is often required by a decision maker.

A vector $u = (u_1, \dots, u_k)$ is said to dominate another vector $v = (v_1, \dots, v_k)$, denoted as $u \prec v$, iff $\forall i \in \{1, \dots, k\}$, $u_i \leq v_i$ and $u \neq v$. A feasible solution $X \in \Omega$ of problem (1) is called *Pareto optimal solution*, iff $\nexists Y \in \Omega$ such that $F(Y) \prec F(X)$. The set of all Pareto optimal solutions is called the Pareto Set, denoted as

$$PS = \{X \in \Omega | \ \nexists Y \in \Omega, F(Y) \prec F(X)\}.$$

The image of the PS in the objective space is called the Pareto Front $PF = \{F(X) | X \in PS\}$.

Multi-Objective Evolutionary Algorithms (**MOEAs**) can obtain an approximate PF of a MOP in a single run by using various operators to iteratively generate a population of such solutions. The aim is to produce a diverse set of non-dominated solutions that is as close as possible to the real PF. Several techniques were proposed for improving their performance, such as nithcing techniques for improving diversity and/or local search methods for improving convergence [2]. MOEAs, such as, NSGA-II [3], SPEA-II [4] are based on Pareto Dominance, while MOEA/D [5] and MOGLS [6] are examples of decompositional MOEAs, relying on conventional aggregation approaches to decompose a MOP into a number of scalar Single-Objective Optimization (SOO) sub-problems.

The hybridization of MOEAs with Local Search (LS) heuristics, also known as Memetic Algorithms (MAs) [7], can provide improved performance, leading to new challenges such as how to select the appropriate LS method within a pool of LS methods in order to identify, in an effective manner, the best local solution within a neighbourhood.

In SOO, an Evolutionary Algorithm is often hybridized with a LS method either randomly or deterministically and can achieve both good exploration and exploitation [8]. The choice of LS is important towards search effectiveness [9], and the wrong choice of local search can have a negative impact [10]. Ong and Keane in [11] propose an adaptive search approach, coined *Meta-Lamarckian learning* MA, that intelligently selects the most suitable LS approach from a pool of LSs for a particular problem during the evolution and define a common reward measure η of the improvements contributed by a LS to a solution that has been searched.

In MOO however, due to the multiple often conflicting objectives involved, things are even more complicated since different LSs can exhibit different biases on the same instance of the same problem domain but for different objectives. The approach of Ong and Keane in [11], cannot be directly applied in MOO and cannot be combined with a MOEA based on Pareto dominance, which tackle a MOP as a whole. In the MOEA/D approach, on the other hand, the MOP is decomposed into a set of SOO subproblems, each one associated to a weight vector, that are solved using SOO techniques and neighbourhood information making the use of Meta-Lamarckian learning for each subproblem possible.

Multi-objective variants of PFSSP have been extensively studied in the literature. Reviews and comparison of various constructive and improvement heuristics based on various metaheuristics such as Genetic Algorithms (GA), Simulated Annealing (SA), Variable Neighborhood Search (VNS) and Tabu Search (TS) can be found in [12], [13]. The first hybrid MOEA was implemented in [6], and improved in [14], as a Multi-Objective Genetic Local Search (MOGLS) approach to solve a multi-objective version of PFSSP, in which the multiple objectives were aggregated into a scalar fitness function using random weights for parent selection and LS. In [15], a hybrid Multi-Objective Particle Swarm Optimization (MOPSO) with Simulated Annealing is proposed. A problem-specific LS based on the NEH-heuristic is first applied to good solutions and an adaptive Meta-Lamarckian learning strategy is employed in order to decide which neighborhood (Swap / Insert / Inverse) will be used each time.

In this paper, we combine Meta-Lamarckian learning with MOEA to study a multi-objective Permutation Flow Shop Scheduling Problem (MO-PPFSP), formulated in Section II. In Section III, we propose a new algorithm, named MOEA/D-SR, which follows the general framework of MOEA/D [5], combined with a pool of generalized LS methods and a Stochastic Roulette-wheel strategy to adaptively learn the effectiveness of each LS, and select the best performing LS for each objective function of each problem instance of each class of problems, online during the evolution. No problem-specific heuristics are used in the design of the proposed approach, allowing its generalizability to other multi-objective combinatorial optimization problems, e.g. the Capacitated VRP [16].

II. Problem Definition and Formulation (MO-PFSSP)

In this section, we formulate the *Multi-Objective Permutation Flow Shop Scheduling Problem* (**MO-PFSSP**).

Given n jobs and a series of m machines, and for $i = 1, ..., n$ and $k = 1, ..., m$, the processing times $P(i,k)$ of job i on machine k and the due dates d_i of job i, let $C(\pi_i, k)$ denote the completion time of job π_i on machine k.

The completion times of a permutation $\pi = (\pi_1, \pi_2, ..., \pi_n)$ i.e., a processing order of the jobs on each machine, can be recursively calculated as follows:

$$\begin{aligned}
&C(\pi_1, 1) = P(\pi_1, 1)\\
&C(\pi_i, 1) = C(\pi_{i-1}, 1) + P(\pi_i, 1), \quad (2 \le i \le n);\\
&C(\pi_1, k) = C(\pi_1, k-1) + P(\pi_1, k), \quad (2 \le k \le m);\\
&C(\pi_i, k) = max\{C(\pi_{i-1}, k), C(\pi_i, k-1)\} + P(\pi_i, k),\\
&\qquad\qquad (2 \le i \le n,\ 2 \le k \le m).
\end{aligned}$$

The lateness of job π_i is defined as $L_{\pi_i} = C(\pi_i, m) - d_{\pi_i}$ and the tardiness of job π_i is defined as $T_{\pi_i} = max\{L_{\pi_i}, 0\}$.

MO-PFSSP can be formulated as follows: determine a permutation of jobs $\pi = (\pi_1, \pi_2, ..., \pi_n)$, which **minimizes** the following two objectives:

- *Objective 1*: **makespan** (or total completion time), that is, the time required to complete the last job π_n on the last machine m, given by $C_{max}(\pi) = C(\pi_n, m)$;
- *Objective 2*: **maximum tardiness**, defined as $$T_{max}(\pi) = max_{1 \le i \le n}\{T_{\pi_i}\}.$$

III. Proposed hybrid MOEA/D Approach with Learning

In this section we present our proposed hybrid Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D), endowed with the Stochastic Roulette-wheel learning strategy, coined **MOEA/D-SR**.

A. MOEA/D extended framework

Given a pool of six general-purpose heuristics we extend the general framework of the conventional MOEA/D [5] in Algorithm 1 in two directions at Step 2.2: i) permit its hybridization with a single local search heuristic from the generic pool, giving rise to six individualistic hybrid MOEA/D's, and ii) design a Meta-Lamarckian learning strategy (Stochastic Roulette-wheel) that adaptively selects at each step of the evolution and for each problem neighbourhood the best performing local search heuristic from the generic pool of local search heuristics, giving rise to MOEA/D-SR.

Algorithm 1 - hybrid MOEA/D with learning

```
Input:
• an instance of MO-PFSSP (see Section II);
• the number $N$ of decomposed subproblems = population size;
• uniformly spread weight vectors $\{\lambda^1, \dots, \lambda^N\}$;
• reward vectors $R^i = (r_1, \dots, r_L)$ assigned to $i^{th}$ subproblem;
• the size of the neighbourhood $T$ of each subproblem;
• tournament size $\tau$, crossover rate $c_r$ and mutation rate $m_r$;
• a termination criterion: max number of generations = $gen^m$;
• the pool $P_{LS}$ of local search heuristics of size $L = 6$.
Output: the external population, $EP$.
Step 0 - Pre-processing:
        Decomposition: into a set of $N$ single-objective subproblems having weights $\{\lambda^1, \dots, \lambda^N\}$ respectively;
        Neighborhoods: Define $B^i$ for the $i^{th}$ subproblem to include the $T$ closest weight vectors of $\lambda^i$.
        Setup: Set $EP := \emptyset$; $gen := 0$; $IP_{gen} := \emptyset$;
Step 1 - Initialization: Set Pareto Front $PF = \emptyset$ and reward vectors $R^i = 0$. For each subproblem, umniformly randomly generate and evaluate an initial internal population $IP_0 = \{X^1, \cdots, X^N\}$. Set $gen = 1$.
Step 2: For $i = 1, \dots N$ do
        Step 2.1 - Genetic Operators: For $i^{th}$ subproblem, generate new solution $Y^i$ using the genetic operators.
        Step 2.2 - Learning + Local Search: Select a local search heuristic from the pool $P_{LS}$ of LS heuristics, based on a learning strategy and then apply the chosen LS heuristic on $Y^i$ to produce $Z^i$.
        Step 2.3 - Update: Update reference point $z^*$ and use $Z^i$ to update $IP_{gen}$, $EP$ and the neighborhood $B^i$ of the $T$ closest neighbor solutions of $Z^i$.
Step 3 - Stopping criterion: If stopping criterion is satisfied, i.e., $gen = gen^m$, then stop and output $EP$, otherwise $gen = gen + 1$, go to Step 2.
```

MOEA/D requires first some pre-processing procedures at **Step 0**, before initiating the main part of the algorithm. The main steps are briefly summarized and discussed next.

Encoding Representation: A solution of MO-PFSSP is represented by a vector of size equal to the number of jobs n, whose components are jobs according to their processing order, that is, job in position i, denoted by π_i, is the i^{th} job to be processed, see Figure 1 for an example.

Decomposition: Initially, the MO-PFSSP is decomposed into a number of N scalar subproblems using the Tchebycheff approach as originally proposed in [5]. Given the objective vector $F(X) = (f_1(X), f_2(X))$, weight vector λ^i, $(1 \leq i \leq N)$, that remains fixed for each subproblem for the whole evolution and a reference point $z^* = (z_1, z_2)$, which is a vector with all the best values z_k found so far for each objective f_k, the objective function of a subproblem i is stated as:

$$g(X|\lambda^i, z^*) = \sum_{k=1}^{2} |\lambda_k^i f_k(X) - z_k|. \qquad (2)$$

Neighbourhood: A neighbourhood (or subpopulation) B^i is maintained for each of the N subproblems associated with weight vector λ^i, composed of the indeces of the subproblems whose associated weight vectors are the T closest (in terms of Euclidean distance) to λ^i. According to Zhang and Li in [5] one expects optimal solutions in neighbouring sub-problems to be close to each other in the search space, so the exchange of genetic information should be helpful.

Step 1 - Initialization: The algorithm commences by creating an initial population $IP_0 = \{X^1, ..., X^N\}$ of solutions one for each subproblem, named Internal Population (IP) of generation $gen = 0$. The initial solutions are randomly generated and each individual is evaluated as described earlier. Set $gen = 1$;

Step 2.1 - Genetic Operation: At each generation gen, for each subproblem i with objective function $g(X^i|\lambda^i, z^*)$, the population IP_{gen} is evolved by generating a new solution Y^i, known as offspring using conventional genetic operators (i.e., Selection, Crossover and Mutation as in [5]). In particular, two parent solutions are randomly selected from the neighbourhood B^i of subproblem i. The two parent solutions are recombined using a two-point crossover to produce a new solution - the offspring - with a probability r_c. The offspring is then modified with a random mutation operator with a probability r_m. Evaluate the new solution Y^i using Eq. (2).

Step 2.2 - Learning + Local Search: At each generation gen, for each subproblem i, a local search heuristic from the pool P_{LS} is adaptively selected based on the assigned reward vector $R^i = (r_1, \dots, r_L)$ and it is used greedily, for a pre-defined number I of iterations, to generate an improved solution Z^i by locally optimizing solution Y^i obtained by the genetic operators. The reward vector R^i is constructed and updated at each generation according to the adopted Meta-Lamarckian learning strategy. The solution Z^i is selected as the new representative of the sub-problem i and therefore replaces the current best solution X^i, iff $g(Z^i|\lambda^i, z^*) < g(X^i|\lambda^i, z^*)$.

Step 2.3 - Update: Use solution Z^i to update the reference point $z*$, the internal population IP, the set of non-dominated solutions PF found so far and neighbourhood B^i of the subproblem i B^i. If $i < N$ then $i = i + 1$ and goto Step 2.1. The same process is followed for all N sub-problems.

Step 3 - Stopping Criteria: If $gen = gen^m$ then terminate the algorithm and output the PF, otherwise goto Step 2.1.

B. Pool of general-purpose Local Search heuristics

In order to maintain the robustness and generalizability of our proposition on the impact of the adopted learning strategy, no problem-specific heuristics are used. Instead, in the pool P_{LS} used in **Step 2.2** of Algorithm 1, we only include six general local search heuristics, frequently used on permutation or sequencing problems [1]:

- **Swap Heuristic** (Sw): randomly selects and swaps the assigned order of two jobs
- **Double Swap** (DSw): performs the swap heuristic twice.
- **Swap Adjacent** (SwA): randomly select a job and swap its assigned processing order with that of its adjacent job.
- **Shift Heuristic** (Sh): randomly chooses to perform backward or forward shift. A backward shift randomly selects a job from its current position i and inserts it at position

j where $i > j$, shifting to the right all values in between. A forward shift is similar to backward shift, but it selects a job from its current position i and inserts it at a position j, where $i < j$, shifting to the left all values in between.

- **Double Shift** (DSh): applies a combination of Backward and Forward Shift.
- **Inverse** (IH): randomly selects a subsequence of jobs in the solution and reversing their order.

C. Definition of Reward Function

Recall that for each sub-problem i, solution X was the best solution found so far during the evolution Y is the offspring of X, obtained by genetic operators, and Z is the new solution obtained by applying a local search heuristic from the pool P_{LS} to Y. During the Meta-Lamarckian learning in Step 2.2 of Algorithm 1 a **reward** (ranging from 0 to 1) is calculated to measure the relative improvement contributed by a local search heuristic using the following rules:

$$r = \begin{cases} 1 & \text{(a)} \quad \text{if } g(Z) < g(X) < g(Y) \\ \frac{g(Y)-g(Z)}{g(X)-g(Z)} & \text{(b)} \quad \text{if } g(Z) < g(Y) \leq g(X) \\ \frac{g(Y)-g(Z)}{g(Y)-g(X)} & \text{(c)} \quad \text{if } g(X) \leq g(Z) < g(Y) \\ 0 & \text{(d)} \quad \text{otherwise} \end{cases} \qquad (3)$$

where $g(X)$, $g(Y)$ and $g(Z)$ correspond to $g(X|\lambda^i, z*)$, $g(Y|\lambda^i, z*)$ and $g(Z|\lambda^i, z*)$, respectively. The proposed reward function measures the contribution of the local search approach in the scalar objective function space by taking into consideration the actual replacement of $g(Z)$ towards the optimal solution with respect to $g(X)$ and $g(Y)$.

D. Stochastic Roulette-wheel (SR) Learning Strategy

In Step 2.2 of Algorithm 1, at the beginning of the evolution and for a predefined number of generations g^t, each single local search heuristic from the pool P_{LS} is given the opportunity to hybridize with the MOEA for locally optimizing the solution Y^i of each subproblem i. The reward of each local search is calculated using Equation 3. Those initial reward values will be used later to guide future LS choices and will change dynamically as the overall search progresses. This is commonly known as the training stage, after which the learning phase takes over, using the **Stochastic Roulette-wheel** (SR) strategy, which proceeds as follows:

Step 1: Sum the reward values $r_j \in R^i$ of all local search approaches for subproblem i.

Step 2: Determine normalized relative reward value of the components of R^i.

Step 3: Assign space on the roulette wheel for each local search based on the normalized value.

Step 4: Generate a random number between 0 and 1, select the local search corresponding to the portion of the wheel in which the chosen random number falls.

The SR strategy ensures that the probability of a LS approach being selected is biased from its own previous performance, which changes dynamically as the overall search progresses. At the same time, it allows for diversity in the choice of LS since it restricts a LS from completely dominating the search.

E. Algorithms: MOEA/D-variants defined

The MOEA/D variants considered in this paper are all based on **Algorithm 1** and differ only in Step 2.2 (Learning + LS).

The conventional **MOEA/D** as proposed by Zhang and Li in [5] applies no local search heuristics (although this hybridization was an optional step). In this case, reward values for all local searches are set to 0 and never get updated.

The hybridization of the conventional MOEA/D with a single selected LS heuristic that is used throughout the search is coined **Individualistic MOEA/D**. In this case, Equation 3 is not used to calculate the reward of each local search, instead, the reward of the selected LS is set to 1, and the rewards of the remaining LSs are set to 0 throughout the evolution. In this paper, six different Individualistic MOEA/D approaches are designed, i.e., **MOEA/D-Sw**, **MOEA/D-SwA**, **MOEA/D-DSw**, **MOEA/D-Sh**, **MOEA/D-DSh** and **MOEA/D-IH**, by hybridizing MOEA/D with swap, swap-adjacent, double swap, shift, double shift and inverse heuristics, respectively.

Finally our proposed method, coined **MOEA/D-SR**, is a decompositional MOEA that uses the Stochastic Roulette-wheel approach as its learning strategy.

IV. EXPERIMENTAL STUDIES

In this section, we introduce the performance metrics utilized to evaluate the algorithms' performance, discuss the test instances of the MO-PFSSP and present the parameter settings of the examined MOEA/D variants. In order to evaluate the efficacy of incorporating the Meta-Lamarckian learning approach in MOEA/D two experimental series are developed:

- **Experimental series 1** compares the conventional MOEA/D approach [5] with the proposed MOEA/D-SR that uses the Stochastic Roulette-wheel learning strategy.
- **Experimental series 2** compares the proposed MOEA/D-SR with the six individualistic MOEA/D variants.

A. Performance Metrics

It is desirable that the obtained non-dominated set of a MOEA is of high quality, that is as close to the true Pareto Front as possible, and distributed as diversely and uniformly as possible. In the literature, there is no single metric that can reflect both of these aspects and thus a number of metrics are often used [2], [17]. In this study, we have used the following three metrics to evaluate our proposed approach:

- **Coverage (C):** commonly used for comparing two sets of non-dominated solutions A and B, the $C(A, B)$ metric calculates the ratio of the non-dominated solutions in B dominated by the non-dominated solutions in A, divided by the total number of non-dominated solutions in B; a higher value for $C(A, B)$ is an indication of higher quality of solutions in A than in B.
- **Distance from reference set (I_D):** shows the average distance from a solution in the reference set R to the

TABLE I

EXPERIMENTAL SERIES 1 - THE PROPOSED MOEA/D-SR IS COMPARED WITH THE CONVENTIONAL MOEA/D IN TERMS OF THE PERFORMANCE METRICS I_D, I_H AND C IN TEST INSTANCES T1-T3. BEST RESULTS OF EACH TEST INSTANCE DENOTED IN BOLD.

Alg:	MOEA/D-SR		MOEA/D		C(MOEA/D-SR,MOEA/D)	C(MOEA/D-SR,MOEA/D)
TI	I_D	I_H	I_D	I_H		
1:	**70.74**	**0.06**	79.66	**0.06**	**0.88**	0.00
2:	**120.25**	**0.10**	258.73	0.04	**1.00**	0.00
3:	**181.63**	**0.12**	419.60	0.04	**0.97**	0.00

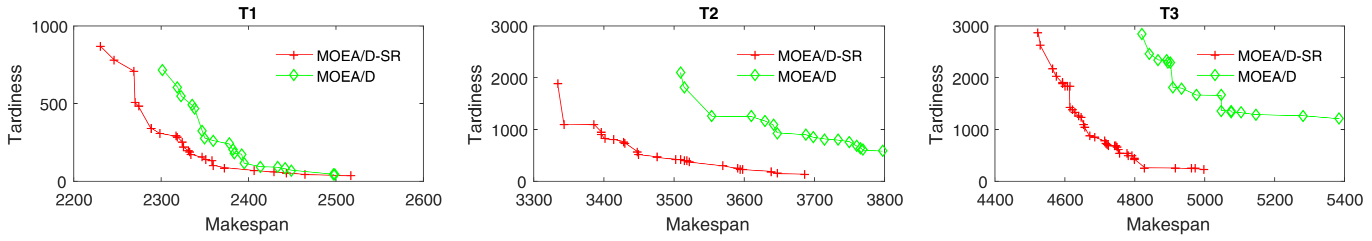

Fig. 2. Experimental Series 1 - Comparison between proposed MOEA/D-SR (with Stochastic Roulette-wheel learning strategy) and conventional MOEA/D.

closest solution in A. The smaller the value of I_D the closer the set A is to R indicating better convergence. In the absence of the real reference set R, the average distance of each single point to the nadir point is used.

- **Hypervolume** (I_H)**:** indicates the area dominated by at least one solution in the obtained non-dominated set A. Therefore high I_H indicates better diversity.

B. Experimental Layout and Algorithmic Settings

We evaluate the performance of MOEA/D-SR on three (n-job, m-machine) benchmark PFSSP test instances T1 ($n = 20, m = 20$), T2 ($n = 40, m = 20$), and T3 ($n = 60, m = 20$), defined in [14] as follows: the processing time of each job on each machine was specified as a random integer in the interval $[1, 99]$ and the due date of each job was specified by adding a random integer in the interval $[-100, 100]$ to its actual completion time in a randomly generated schedule.

The algorithmic parameters are set as follows: termination criterion gen^m=1000, population size and number of subproblems N=500, tournament size τ=10, crossover rate r_c=0.8, mutation rate r_m=0.2, neighbourhood size T=14, the size of the pool P_{LS} of local search heuristics L=6, the number of iterations for each local search is set to I=10 and the training phase for the MOEA/D with Meta-Lamarckian Learning strategy is set to g^t=100. Note that in our experimental studies we have used the same number of function evaluations for all methods, for fairness. The makespan and maximum tardiness objectives are evaluated as in Section II. All algorithms were coded in Java programming language and run on an Intel(R) Core(M) i5 CPU 2.4GHz Windows 7 server with 4 GB RAM.

C. Experimental Results on MO-PFSSP

The results of the experimental series are presented below.

Exp. Series 1- **MOEA/D-SR vs conventional MOEA/D**: Figure 2 clearly shows that the proposed hybrid MOEA/D-SR with Stochastic Roulette-wheel learning strategy improves the performance of the conventional MOEA/D in terms of both convergence and diversity in all test instances, even more so when the problem becomes more difficult as the number of jobs n increases. This is also evident in Table I that summarizes the statistical performance of the two algorithms in terms of the considered metrics. The proposed MOEA/D-SR technique provides a more diverse (i.e., high I_H) Pareto Front, that is closer to the reference set (i.e., low I_D) and of higher quality (i.e., high C) compared to individualistic MOEA/D variants, in all test instances.

Exp. Series 2 - **MOEA/D-SR vs individualistic MOEA/Ds**: In this experimental series, the proposed hybrid MOEA/D-SR with the Stochastic Roulette-wheel approach is compared with the six individualistic MOEA/D variants defined above. Figure 3 shows that in test instances T2-T3, MOEA/D-SR outperforms its competitors in terms of both convergence and diversity. In particular, MOEA/D-SR has obtained a PF that dominates most of the non-dominated solutions obtained by the individualistic MOEA/Ds providing a better approximation towards the nadir point. The results in Table II show that MOEA/D-SR provides better I_D than all of its competitors, except in T1 and T2 where MOEA/D-DSh obtains slightly better results. In terms of diversity I_H, it performs no worse than all other individualistic MOEA/Ds, except in T3 where MOEA/D-Sh is slightly better. In terms of the Coverage metric C, we observe that non-dominated solutions obtained by MOEA/D-SR dominate most (on average 80%) of the non-dominated solutions obtained by individualistic MOEA/D variants in all test instances. The non-dominated solutions obtained by MOEA/D-SR are dominated, on average, by 13% and 7% only by the non-dominated solutions obtained by individualistic MOEA/D-Sh and MOEA/D-DSh, respectively.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we study an important multi-objective scheduling problem (MO-PFSSP) which aims to find a permutation of n jobs to be processed sequentially on m machines, so as

TABLE II
RESULTS OF EXPERIMENTAL SERIES 2 - COMPARISON BETWEEN THE MOEA/D-SR AND ALL INDIVIDUALISTIC MOEA/D VARIANTS IN TERMS OF THE PERFORMANCE METRICS I_D, I_H AND C. THE BEST RESULTS OF EACH TEST INSTANCE ARE DENOTED IN BOLD.

Alg:	M-SR		M-Sw		M-SwA		M-DSw		M-Sh		M-DSh		M-IH	
TI	I_D	I_H	I_D	I_H	I_D	I_H	I_D	I_H	I_D	I_H	I_D	I_H	I_D	I_H
1:	62.14	**0.10**	71.34	0.09	95.51	0.07	83.46	0.08	65.01	0.09	**61.93**	**0.10**	88.80	0.09
2:	115.03	**0.11**	171.28	0.06	210.14	0.06	170.15	0.07	140.06	0.09	**103.11**	0.08	219.28	0.06
3:	**213.18**	0.08	348.21	0.06	445.13	0.04	271.50	0.05	260.37	**0.09**	253.63	0.08	387.87	0.07

TI	C(SR,Sw)	C(Sw,SR)	C(SR,SwA)	C(SwA,SR)	C(SR,DSw)	C(DSw,SR)	C(SR,Sh)	C(Sh,SR)	C(SR,DSh)	C(DSh,SR)	C(SR,IH)	C(IH,SR)
1:	**0.72**	0.00	**0.92**	0.00	**0.68**	0.00	**0.44**	0.09	**0.40**	0.20	**0.56**	0.00
2:	**0.92**	0.00	**0.96**	0.00	**1.00**	0.00	**0.81**	0.17	**0.96**	0.00	**0.96**	0.00
3:	**0.81**	0.00	**0.97**	0.00	**0.95**	0.00	**0.62**	0.14	**0.92**	0.00	**0.89**	0.00

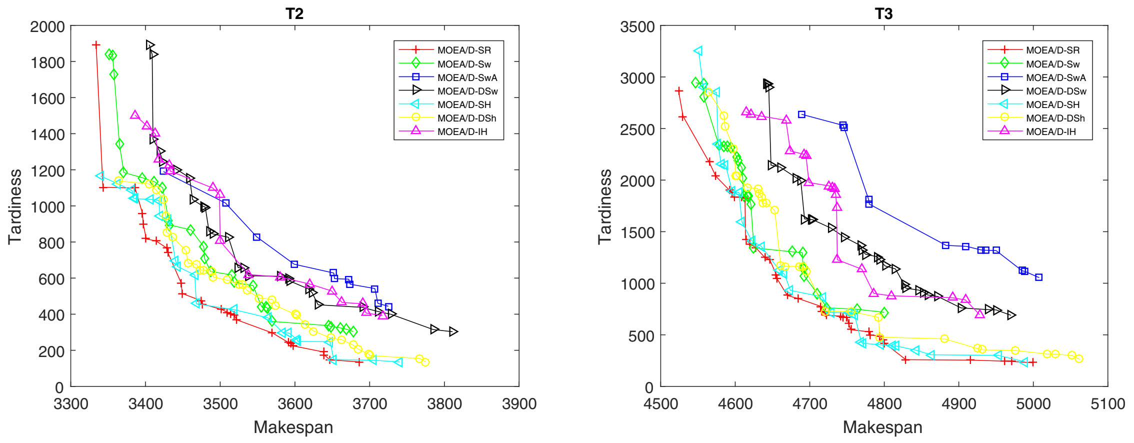

Fig. 3. Experimental Series 2 - Comparison between MOEA/D-SR (with Stochastic Roulette-wheel learning strategy) and Individualistic MOEA/D variants.

to minimize simultaneously the makespan and maximum tardiness objectives. Our proposed MOEA/D-SR algorithm, follows the general framework of a Multi-Objective Evolutionary Algorithm based on Decomposition, combined with a learning strategy (Stochastic Roulette-wheel approach), for adaptively selecting from a generic pool of local search heuristics the best performing one, based on the problem's properties and objective functions, so as to locally optimize the solutions during the evolution. We evaluate our algorithm on various benchmark problems and the experimental results show that MOEA/D-SR successfully learns during the evolution and provides a more diverse and high quality set of Pareto-optimal solutions compared to its competitors.

In the future, we consider designing problem-specific local search heuristics that can be incorporated in a hybrid MOEA/D, investigate more learning strategies and extend our study to other real-life combinatorial MOO problems.

REFERENCES

[1] P. Brucker, *Scheduling Algorithms*. Springer, 2007.
[2] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.
[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
[4] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, Athens, Greece, 2002, pp. 95–100.
[5] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.
[6] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, no. 3, pp. 392–403, 1998.
[7] P. Moscato and C. Cotta, *A Gentle Introduction to Memetic Algorithms*. Springer, 2003, vol. 57, ch. 5, pp. 105–144.
[8] N. Krasnogor, "Studies on the theory and design space of memetic algorithms," Ph.D. dissertation, University of the West of England, 2002.
[9] C. R. Reeves, "Statistical properties of combinatorial landscapes: An application to scheduling problems," in *MIC2001: Proceedings of the 4th Metaheuristic International Conference*, 2001, pp. 691–695.
[10] L. Davis, *Handbook of Genetic Algorithms*. V. N. Reinhold, 1991.
[11] Y. S. Ong and A. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Tr. on Evol. Comp.*, vol. 8, no. 2, pp. 99–110, April 2004.
[12] M. S. N. Fernando Luis Rossi and R. F. T. Neto, "Evaluation of high performance constructive heuristics for the flow shop with makespan minimization," *The International Journal of Advanced Manufacturing Technology*, vol. 87, no. 1, pp. 125–136, 2016.
[13] M. C. G. Minella, R. Ruiz, "A review and evaluation of multiobjective algorithms for the flowshop scheduling problem," *INFORMS Journal on Computing*, vol. 20, pp. 451–471, 2008.
[14] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.
[15] B.-B. Li, L. Wang, and B. Liu, "An effective pso-based hybrid algorithm for multiobjective permutation flow shop scheduling," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 38, pp. 818–831, 2008.
[16] A. Konstantinidis, S. Pericleous, and C. Charalambous, "Adaptive evolutionary algorithm for a multi-objective VRP," *International Journal on Engineering Intelligent Systems*, vol. 22, no. 3/4, 2014.
[17] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Tr. on Evol. Comp.*, vol. 7, no. 2, pp. 117–132, 2003.