A discrete differential evolution algorithm for multi-objective permutation flowshop scheduling

M. Baioletti, A. Milani, V. Santucci

Dipartimento di Matematica e Informatica
Università degli Studi di Perugia
Via Vanvitelli, 1
Perugia, Italy

Abstract. Real-world versions of the permutation flowshop scheduling problem (PFSP) have a variety of objective criteria to be optimized simultaneously. Multi-objective PFSP is also a relevant combinatorial multi-objective optimization problem. In this paper we propose a multi-objective evolutionary algorithm for PFSPs by extending the previously proposed discrete differential evolution scheme for single-objective PFSPs. The novelties of this proposal reside on the management of the evolved Pareto front and on the selection operator. A preliminary experimental evaluation has been conducted on three bi-objective PFSPs resulting from all the possible bi-objective combinations of the criteria makespan, total flowtime and total tardiness.

Introduction

The Permutation Flowshop Scheduling Problem (PFSP) is an important type of scheduling problem which has many applications in manufacturing and large scale product fabrication. In this problem there are n jobs $J_1, \ldots, J_n$ and m machines $M_1, \ldots, M_m$. Each job J_i is composed by m operations $O_{i1}, \ldots, O_{im}$. The generic operation O_{ij} can be executed only by the machine M_j and its given processing time is p_{ij}. Moreover, the execution of any operation cannot be interrupted (no pre-emption) and job passing is not allowed, i.e., the jobs must be executed using the same order in every machine. The goal of PFSP is to find the optimal job permutation $\pi = \langle \pi(1), \ldots, \pi(n) \rangle$ with respect to a given objective function. Three important criteria are to minimize the total flowtime (TFT), the makespan (MS) and the total tardiness (TT) defined as follows:

$$TFT(\pi) = \sum_{i=1}^{n} C(\pi(i), m) \tag{1}$$

$$MS(\pi) = \max_{i=1,\ldots,n} C(\pi(i), m) = C(\pi(n), m) \tag{2}$$

$$TT(\pi) = \sum_{i=1}^{n} \max\{C(\pi(i), m) - d_{\pi(i)}, 0\} \tag{3}$$

where $C(h, j)$ is the completion time of the operation O_{hj} and is computed by the following recursive equation

$$C(\pi(i), j) = p_{\pi(i),j} + \max\{C(\pi(i-1), j), C(\pi(i), j-1)\}$$

for $i, j \geq 1$, while the terminal cases are $C(\pi(0), j) = C(i, 0) = 0$. In equation (3), for each job h, also a given delivery date d_h is considered.

The minimization of each one of these criteria is computationally hard. Indeed, both the TFT and TT problems are NP-hard for $m \geq 2$, while the MS minimization becomes NP-hard when $m > 2$.

Many single-optimization algorithms exists, either exact or approximate, for instance: heuristic techniques, local searches or evolutionary algorithms [2]. Anyway, in this paper we investigate the PFSP problem as a multi-objective optimization problem, in which the goal is to find a set of job permutations which are good enough with respect to two or more contrasting criteria, i.e. a set of Pareto optimal solutions.

Given k objective functions $f_1, \ldots, f_k$, a solution x dominates a solution x' (denoted by $x \prec x'$) if $f_i(x) \leq f_i(x')$ for $i = 1, \ldots, k$, and there exists at least an index $j \in \{1, \ldots, k\}$ such that $f_j(x) < f_j(x')$. A solution x is Pareto optimal if there exist no other solution x' such that $x' \prec x$. The Pareto optimal set is the set of all the Pareto optimal solution. If two solutions x and x' are such that neither $x \prec x'$ nor $x' \prec x$, then x and x' are incomparable.

Since the Pareto set is in general very large, the goal is to find an approximation of this set, i.e., a set composed by incomparable solutions which is as close as possible to the Pareto optimal set. One of the most promising approaches to solve multi-objective optimization problems is to use evolutionary algorithms [1].

In the context of multi-objective PFSP, many approaches have been proposed. The surveys [3, 7] describe and compare many algorithms for PFSP with all the three possible combinations of two objectives among TFT, MS and TT.

In this paper we describe an algorithm for multi-objective optimization which is based on Differential Evolution for Permutation (DEP) [6]. DEP is a discrete differential evolution algorithm which directly operates on the permutations space and hence is well suited for permutation optimization problems like PFSP. Indeed, in [6] and in [5], it was shown that DEP reaches state-of-the-art results with respect to total flowtime and makespan single objective optimization. Here, DEP has been extended in order to handle multi-objective problems. The preliminary experimental results show that its performances are comparable with state-of-the-art algorithms.

The rest of the paper is organized as follows. The second section describes the classical Differential Evolution algorithm. Its extension to the permutations space and multi-objective PFSP is introduced in the third section. An experimental investigation of the proposed approach is provided in the fourth section, while conclusions are drawn at the end of the paper.

The Differential Evolution algorithm

In this section we provide a short introduction to Differential Evolution (DE) algorithm. For more detail see [4]. Differential Evolution (DE) is a powerful population-based evolutionary algorithm for optimizing non-linear and even non-differentiable real functions

in $\mathcal{R}^n$. The main peculiarity of DE is to exploit the distribution of the solutions' differences in order to probe the search space.

DE initially generates a random population of NP candidate solutions $x_1, \ldots, x_{NP}$ uniformly distributed in the solutions space. At each generation, DE performs mutation and crossover in order to produce a trial vector u_i for each individual x_i, called target vector, in the current population. Each target vector is then replaced in the next generation by the associated trial vector if and only if the produced trial is fitter than the target. This process is iteratively repeated until a stop criterion is met (e.g., a given amount of fitness evaluations has been performed).

The differential mutation is the core operator of DE and generates a mutant vector v_i for each target individual x_i. The most used mutation scheme is "rand/1" and it is defined as follows:

$$v_i = x_{r_0} + F \cdot (x_{r_1} - x_{r_2}) \tag{4}$$

where r_0, r_1, r_2 are three random integers in $[1, NP]$ mutually different among them. x_{r_0} is called base vector, $x_{r_1} - x_{r_2}$ is the difference vector, and $F > 0$ is the scale factor parameter.In [4] it is argued that the differential mutation confers to DE the ability to automatically adapt the mutation step size and orientation to the fitness landscape at hand.

After the mutation, a crossover operator generates a population of NP trial vectors, i.e. u_i, by recombining each pair composed by the generated mutant v_i and its corresponding target x_i. The most used crossover operator is the binomial one that builds the trial vector u_i taking some components from x_i and some other ones from v_i according to the crossover probability $CR \in [0, 1]$.

Finally, in the selection phase, the next generation population is selected by a one-to-one tournament among x_i and u_i for $1 \leq i \leq NP$.

Discrete Differential Evolution for Multi-Objective Optimization

In this section we describe the proposed Multi-Objective Differential Evolution for Permutation (MODEP) which directly evolves a population of NP permutations $\pi_1, \ldots, \pi_{NP}$. With respect to the classical DE, important variations have been made to the genetic operators of mutation, crossover and selection. Moreover, an additional archive of solutions is introduced to maintain the evolved Pareto front.

To simplify our description, let us restrict to the case of two objective functions f_1 and f_2. A population of NP permutations $\pi_1, \ldots, \pi_{NP}$ is randomly generated at the beginning. At each iteration, a secondary population of trial elements $\upsilon_1, \ldots, \upsilon_{NP}$ is generated by means of the mutation and crossover operators. Then, a selection operator selects, for $i = 1, \ldots, NP$, which element among υ_i and π_i should be part of the population for the next iteration.

The pseudo-code of MODEP is depicted in Alg. 1.

Differential Mutation

The mutation operator used is the same of DEP [6]. It produces a mutant ν_i for each population element π_i using some algebraic concepts related to the symmetric group of permutations. Here we briefly recall its structure:

Algorithm 1 MODEP

```
Initialize Population
Update ND
while num_fit_eval ≤ max_fit_eval do
    for i ← 1 to NP do
        ν_i ← DifferentialMutation(i)
        v_i^(1), v_i^(2) ← Crossover(π_i, ν_i)
        Update ND
        υ_i ← SelectChild(v_i^(1), v_i^(2))
    end for
    for i ← 1 to NP do
        π_i ← Selection (π_i, υ_i)
    end for
end while
```

```
Find r_0, r_1, r_2 different to i and to each other
δ ← π_{r_2}^{-1} ∘ π_{r_1}
S ← RandBS(δ) (S is a sequence of adjacent swaps)
L ← Length(S)
k ← ⌈F · L⌉
ν_i ← π_{r_0}
for j = 1, ..., k apply S_j to ν_i
```

where $\circ$ is the ordinary permutation composition operator, $\cdot^{-1}$ denotes the inverse of a permutation, and $RandBS$ is the randomized bubble sort procedure which allows to decompose a permutation in a sequence of adjacent swaps (that are themselves simple permutations). For more details, see [6].

It is worth to notice that this operator works directly with permutations, simulating from an algebraic point of view, the expression of equation (4).

Crossover

The crossover operator for permutation representations is the same of DEP and produces two children $v_i^{(1)}$ and $v_i^{(2)}$ from π_i and ν_i. The details are described in [6].

The two permutations $v_i^{(1)}$ and $v_i^{(2)}$ are compared with respect to both f_1 and f_2. If $v_i^{(1)}$ dominates $v_i^{(2)}$, then the trial υ_i is $v_i^{(1)}$. Analogously, if $v_i^{(2)}$ dominates $v_i^{(1)}$, then the trial υ_i is $v_i^{(2)}$. When $v_i^{(1)}$ and $v_i^{(2)}$ are incomparable, then one of them is randomly selected to become the trial υ_i.

Selection

The selection operator chooses the new population element π_i' between the old element π_i and the trial ν_i. If $\pi_i \prec \nu_i$, then π_i' becomes π_i, i.e., π_i remains in the population. Otherwise, if $\nu_i \prec \pi_i$ or it is equal to π_i, then π_i' becomes ν_i, that is ν_i replaces π_i in

the next generation population. However, if π_i and ν_i are incomparable, then we use a probabilistic method somehow similar to the α-selection described in [6].

Suppose first that $f_1(\nu_i) < f_1(\pi_i)$ but $f_2(\nu_i) \geq f_2(\pi_i)$. Then, π'_i becomes ν_i with probability $\max\{0, \alpha_2 - \Delta_i^{(2)}\}$, otherwise it retains the old element π_i, where

$$\Delta_i^{(2)} = \frac{f_2(\nu_i) - f_2(\pi_i)}{f_2(\pi_i)}$$

is the relative worsening of ν_i with respect to π_i according to f_2.

Analogously, if $f_1(\nu_i) \geq f_1(\pi_i)$ and $f_2(\nu_i) < f_2(\pi_i)$. Then, π'_i becomes ν_i with probability $\max(0, \alpha_1 - \Delta_i^{(1)})$, where

$$\Delta_i^{(1)} = \frac{f_1(\nu_i) - f_1(\pi_i)}{f_1(\pi_i)}.$$

The rationale behind this selection operator is that ν_i enters the population if it dominates or is equal to π_i or, with a small probability, if it is not too worse than π_i in one of the objective functions, while it is better than π_i in the other objective function. Moreover, note that the probability of accepting a slightly worsening population element linearly shades from α_h, when $\Delta_i^{(h)} = 0$, to 0, when $\Delta_i^{(h)} = \alpha_h$, for $h = 1, 2$.

Therefore, the parameters α_h regulates how worse ν_i can be in order to be accepted in the new population: if $\alpha_1 = \alpha_2 = 0$ only better elements (in the Pareto sense) can replace old elements in the population.

Pareto Front

The algorithm keeps updated the approximated Pareto front ND, which contains all the non-dominated elements ever generated and evaluated. Initially ND contains all the non-dominated population elements created during the random initialization. Then, at each generation, all the couples of children $v_i^{(1)}$ and $v_i^{(2)}$ are used to update ND. A new element v enters ND if it is not dominated by any element of ND. Moreover, all the elements of ND which are dominated by v are removed.

Experimental Results

In this section we report some preliminary experimental results obtained with an implementation of MODEP.

The experiments have been performed by solving the well known Taillard's instances with the additional due times given in [3]. These instances are divided in 11 groups of 10 instances with the same values of n and m. The values of n are in the set $\{20, 50, 100, 200\}$, while m lies in $\{5, 10, 20\}$. The combination $(n = 200, m = 5)$ is not considered. The processing time p_{ij} of each instance are randomly generated in $\{1, \ldots, 99\}$, while the due date of each job J_i are generated by multiplying the value $\sum_{j=1}^{m} p_{ij}$ for a random factor in $[1, 4]$. MODEP has been run 10 times for each instance and the adopted stopping criterion is the maximum number of evaluations, which

has been set to $2000 \cdot n \cdot m$. Three combinations of objectives have been considered: (MS, TFT), (MS, TT), and (TFT, TT). For each execution the obtained Pareto front (corresponding to ND) has been analyzed by computing two performance indices: the hypervolume I_H and the unary multiplicative epsilon I_ϵ^1. I_H is computed as the area delimited by the solutions of ND and a reference point. I_ϵ^1 compares ND with the best known Pareto front B and is computed as

$$I_\epsilon^1 = \max_{x \in B} \min_{y \in ND} \max_{j=1,2} \frac{f_j(y)}{f_j(x)}.$$

The indices have been computed by averaging over the multiple executions and instances for every combination of $n \times m$.

The value for the parameter NP has been set to 100 after some preliminary experiments. The parameter F used in the mutation operator is, as in [6], self-adapted during the evolution. Instead, the values for the selection parameters α_1 and α_2 have been set after a calibration phase according to Table 1.

Table 1. Calibration values for α_1 and α_2

Opt.	$\boldsymbol{\alpha_1}$	$\boldsymbol{\alpha_2}$
(MS, TFT)	0.025	0.015
(MS, TT)	0.01	0.01
(TFT, TT)	0.01	0.01

The results of the optimization of (MS, TFT) are shown in Table 2. MODEP works well on this problem and the values of the second index I_ϵ^1 (whose optimal value is 1) are quite good, while the values for I_H (whose optimal value is 1.44) are however good, compared to those reported in [3].It is worth to notice that, fixing n, I_H seems to have a decreasing behavior as m increases (except when $n = 20$).

Table 2. Results for (MS, TFT)

$\boldsymbol{n}$	$\boldsymbol{m}$	$\boldsymbol{I_H}$	$\boldsymbol{I_\epsilon^1}$
20	5	1.089	1.015
20	10	1.185	1.014
20	20	1.188	1.013
50	5	1.248	1.045
50	10	1.149	1.050
50	20	1.119	1.042
100	5	1.238	1.065
100	10	1.140	1.072
100	20	1.067	1.057
200	10	1.143	1.083
200	20	1.058	1.073

The results of the optimization of (MS, TT) are shown in Table 3 and are similar to those for (MS, TFT), even if the decreasing behavior of I_H with respect to m is not so apparent.

Table 3. Results for (MS, TT)

n	m	I_H	I_ϵ^1
20	5	1.241	1.048
20	10	1.136	1.162
20	20	1.071	1.038
50	5	1.219	1.078
50	10	1.140	1.175
50	20	1.195	1.237
100	5	1.221	1.086
100	10	1.137	1.119
100	20	1.138	1.167
200	10	1.138	1.105
200	20	0.976	1.117

Finally, the results of the optimization of (TFT, TT) are shown in Table 4. Here, while the performances as measured by I_ϵ^1 are still satisfactory, the results of I_H are slightly worse than in the previous cases.

Table 4. Results for (TFT, TT)

n	m	I_H	I_ϵ^1
20	5	1.081	1.026
20	10	1.088	1.193
20	20	1.032	1.038
50	5	0.6525	1.024
50	10	0.899	1.083
50	20	1.021	1.206
100	5	0.540	1.027
100	10	0.660	1.055
100	20	0.799	1.103
200	10	0.588	1.051
200	20	0.641	1.077

Conclusion and Future Work

In this paper we have described an algorithm for optimization of multi-objective permutation flowshop scheduling problems. Some preliminary experimental results show that this approach is promising and reaches results which are comparable to the state-of-the-art algorithms. As a future line of research, we would like to add to our algorithm

some method to enhance the diversity of the population, as done in other evolutionary multi-objective algorithms, like crowding distance or niching techniques.

References

1. Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler. Evolutionary multi-objective optimization. *European Journal of Operational Research*, 181(3):1617–1619, 2007.
2. J. Gupta and J.E. Stafford. Flowshop scheduling research after five decades. *European Journal of Operational Research*, (169):699–711, 2006.
3. Gerardo Minella, Rubén Ruiz, and Michele Ciavotta. A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451–471, 2008.
4. K.V. Price, R.M. Storn, and J.A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Berlin, 2005.
5. Valentino Santucci, Marco Baioletti, and Alfredo Milani. Solving permutation flowshop scheduling problems with a discrete differential evolution algorithm. *submitted to AI Communication*.
6. Valentino Santucci, Marco Baioletti, and Alfredo Milani. A differential evolution algorithm for the permutation flowshop scheduling problem with total flow time criterion. In *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, pages 161–170, 2014.
7. M.M. Yenisey and B. Yagmahan. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, (45):119–135, 2014.