A Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem with Total Flow Time Criterion

Valentino Santucci, Marco Baioletti, and Alfredo Milani

Department of Mathematics and Computer Science
University of Perugia, Italy
{valentino.santucci,baioletti,milani}@dmi.unipg.it

Abstract. In this paper a new discrete Differential Evolution algorithm for the Permutation Flowshop Scheduling Problem with the total flow-time criterion is proposed. The core of the algorithm is the distance-based differential mutation operator defined by means of a new randomized bubble sort algorithm. This mutation scheme allows the Differential Evolution to directly navigate the permutations search space. Experiments were held on a well known benchmark suite and the results show that our proposal outperforms state-of-the-art algorithms on the majority of the problems.

Keywords: Differential Evolution, Permutation Flowshop Scheduling Problem, Randomized Bubble Sort

1 Introduction and Related Works

The Permutation Flowshop Scheduling Problem (PFSP) is a type of scheduling problem widely encountered in areas such as manufacturing and large scale products fabrication [?]. The goal of PFSP is to determine the best permutation $\pi = \langle \pi[1], \ldots, \pi[n] \rangle$ of n jobs that have to be processed through a sequence of m machines.

Here we focus on the Total Flow Time (TFT) criterion that consists in minimizing the objective function

$$f(\pi) = \sum_{j=1}^{n} c(m, \pi[j]) \tag{1}$$

where $c(i, \pi[j])$ is the completion time of job $\pi[j]$ on machine i and is recursively calculated in terms of the processing times $p_{i,\pi[j]}$ as:

$$c(i, \pi[j]) = \begin{cases} p_{i,\pi[j]} & \text{if } i = j = 1 \\ p_{i,\pi[j]} + c(i, \pi[j-1]) & \text{if } i = 1 \text{ and } j > 1 \\ p_{i,\pi[j]} + c(i-1, \pi[j]) & \text{if } i > 1 \text{ and } j = 1 \\ p_{i,\pi[j]} + \max\{c(i, \pi[j-1]), c(i-1, \pi[j])\} & \text{if } i > 1 \text{ and } j > 1 \end{cases} \tag{2}$$

The PFSP with the TFT criterion has been demonstrated to be NP hard for two or more machines. Therefore, even due to its practical interest, many researches have been devoted to finding high quality and near optimal solutions by means of heuristic or meta-heuristic approaches [?,?]. A report of the state-of-the-art methods for PFSP-TFT has been recently provided in [?] where it is shown that the most performing meta-heuristics are: VNS$_4$ [?], AGA [?] and GM-EDA together with its hybrid variant HGM-EDA [?]. VNS$_4$ applies a variable neighborhood search (VNS) to an initial permutation built by means of a constructive heuristic called LR(n/m) [?]. AGA is an asynchronous genetic algorithm hybridized with VNS. GM-EDA is an estimation of distribution algorithm that adopts a probabilistic model for the permutations space known as generalized Mallows model, while HGM-EDA represents the hybridization of GM-EDA with a VNS scheme.

Differential Evolution [?] is one of the many approaches to evolutionary computation [?,?,?]. Although its effectiveness in numerical spaces, DE applications to combinatorial problems, and in particular to permutation-based problems, are still unsatisfactory. To the best of our knowledge, all the DE algorithms for the PFSP proposed in literature (see for example the schemes reported in [?] or the more recent ones [?,?]) adopt some transformation scheme to encode permutations into numerical vectors. This distinction between the phenotypic and genotypic space introduces large plateaus in the numerical landscape and is probably the reason of their poor performances. To address this issue, in this paper we propose a discrete DE scheme for the PFSP-TFT problem that works directly on the permutations space. Since the differential mutation operator has been generally considered the key component of DE [?], our approach mainly relies on a differential mutation operator that directly handles permutations, thus trying to fruitfully bring the DE search properties from the numerical space to the combinatorial space of permutations. Furthermore, a new $O(n^2)$ randomized bubble sort algorithm is provided.

The rest of the paper is organized as follows. The permutation-based differential mutation operator and the new randomized bubble sort algorithm are introduced and motivated in Section **??**. The full DE scheme for PFSP-TFT is described in Section **??**. An experimental analysis of the proposed approach is provided in Section **??**. Finally, conclusions are drawn in Section **??** and some future lines of research are depicted.

2 Differential Mutation in the Permutations Space

Differential Evolution (DE) [?] is a popular and powerful evolutionary algorithm over continuous search spaces using the differential mutation operator as its key component [?]. In the most common variant, for each population individual $x_i \in \mathbb{R}^n$, three different parents $x_{r_0}, x_{r_1}, x_{r_2}$ are randomly selected from the current population and a mutant $v_i \in \mathbb{R}^n$ is generated according to

$$v_i = x_{r_0} + F \cdot (x_{r_1} - x_{r_2}) \tag{3}$$

where the scalar parameter F usually lies in $(0, 1]$. It has been argued that the differential mutation confers to DE the "contour matching" property (term coined by Price et al. in [?]), i.e., it allows DE to automatically adapt both mutation step size and orientation to the objective function landscape.

Here we propose a differential mutation scheme that directly works on the permutations space and that inherits, in some geometric sense, the "contour matching" property of its numerical counterpart.

The permutations of the set $\{1, 2, \ldots, n\}$, together with the usual permutations composition operator $\circ$, form a group denoted by $S(n)$ where each $\pi \in S(n)$ has its inverse, denoted by π^{-1}.

It is possible to bring the classical concepts of sum and difference of $\mathbb{R}^n$ in $S(n)$. Indeed, by defining the sum of $\pi_1, \pi_2 \in S(n)$ as $\pi_1 \circ \pi_2$, their difference can be straightforwardly defined as $\pi_2^{-1} \circ \pi_1$ since $\pi_1 = \pi_2 \circ \left(\pi_2^{-1} \circ \pi_1\right)$. Therefore, by temporarily omitting the scale factor F, equation (**??**) can be rewritten for permutations as:

$$\nu_i = \pi_{r_0} \circ \left(\pi_{r_2}^{-1} \circ \pi_{r_1}\right) \tag{4}$$

In order to introduce the scale factor F in equation (**??**) we need to define an operation which, given a scalar $F \in [0, 1]$, scales down a permutation π to a "truncated" permutation $F \cdot \pi$. A possible approach is to choose a set of generators $G \subseteq S(n)$ and decompose π in the compositions chain $g_1 \circ \cdots \circ g_L$ where $g_1, \ldots, g_L \in G$. Therefore, by defining $F \cdot \pi = g_1 \circ \cdots \circ g_k$, with $k = \lceil F \cdot L \rceil$, it is finally possible to provide a differential mutation for permutations as:

$$\nu_i = \pi_{r_0} \circ \left(F \cdot \left(\pi_{r_2}^{-1} \circ \pi_{r_1}\right)\right) \tag{5}$$

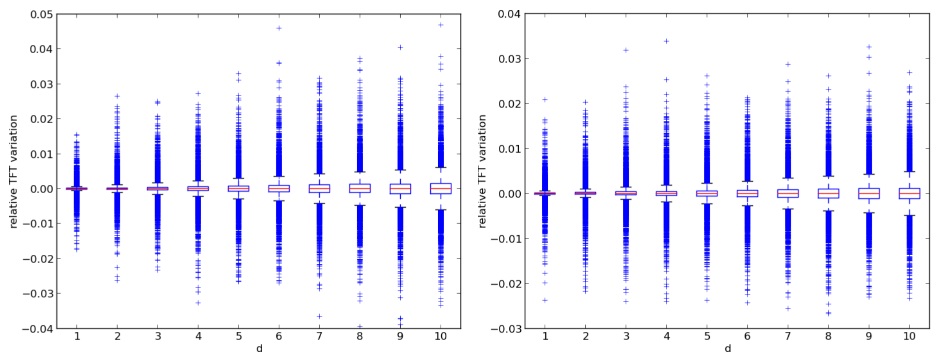
Interestingly, the introduction of a generators set G allows a useful geometric interpretation of the search space. Indeed, given $G \subseteq S(n)$, it is possible to represents the permutations search space as a Cayley graph Γ, i.e., a regular graph whose vertices are the permutations of $S(n)$ and, for any $\pi \in S(n)$ and $g \in G$, the vertices corresponding to π and $\pi \circ g$ are joined by an edge labeled with g. This allows in turn: (1) to derive a metric distance function corresponding to the length of a shortest path between two permutations in Γ, (2) to view the difference between π_1 and π_2 as the compositions chain of the edges labels in a shortest path from π_2 to π_1 in Γ, and (3) to interpret the scaled difference as a truncated shortest path.

However, different sets of generators are possible for $S(n)$. Each one may lead to a different search space structure thus have a different impact on the search algorithm. Here, we consider the three main generators sets of $S(n)$ [?]:

- the set of all transpositions $T = \{(i, j)_T : 1 \le i < j \le n\}$, where $(i, j)_T$ denotes the permutation which only swaps the elements at places i and j,
- the set of all insertions $I = \{(i, j)_I : i \ne j \text{ and } 1 \le i, j \le n\}$, where $(i, j)_I$ denotes the permutation that shifts the element at place j to place i,
- the set of all simple transpositions $ST = \{(i, i{+}1)_T : 1 \le i \le n - 1\}$, i.e., the permutations which only swap two adjacent elements (note that $(i, i{+}1)_T = (i, i{+}1)_I = (i{+}1, i)_I$).

T and I have respectively $\binom{n}{2}$ and $(n-1)^2$ elements, and both produce a search space diameter of $n-1$. Their induced distance functions are known in literature as, respectively, Cayley distance and Ulam distance [?]. Instead, ST, which is a proper subset of both T and I, has $n-1$ elements and provides a diameter of $\binom{n}{2}$. Its induced distance function is known as Kendall-τ distance [?] and equals the number of inversions of either $\pi_2^{-1} \circ \pi_1$ or $\pi_1^{-1} \circ \pi_2$.

The choice among these sets of generators has been made by exploiting the hypothesis that a smoother objective function landscape is a benefit for an evolutionary algorithm. In order to detect which among T, I and ST produces the smoothest landscape on the PFSP-TFT problem, we made two experimental investigations. For several instances of the Taillard benchmark problems (see Section **??**) we have generated 10 000 random permutations. For each one, and for $d = 1, \dots, 10$, we have tabulated its TFT relative difference after the application of a random transposition of the type $(i, i{+}d)_T$ (first experiment) and a random insertion of the type $(i, i{+}d)_I$ (second experiment). From the box-plots reported in Figures **??** and **??** for the first instance of the Taillard problems 100×5 (other instances have the same behavior) it is possible to deduce that $d = 1$ provides the smoother TFT variation and that this variation increases with d. Therefore, by recalling the fact that both transpositions and insertions reduce to simple transpositions when $d = 1$, the search space for the differential mutation operator has been structured using ST as set of generators.



(a) using T (b) using I

Fig. 2: Relative TFT variations on the first instance of the 100×5 Taillard problems

The truncated permutation $F \cdot \pi$ can be computed using the well known bubble sort algorithm. However, $F \cdot \pi$ is not unique in general because π can have several different shortest representations as compositions chain of simple transpositions. Hence, in order to design a mutation scheme as fair as possible, we propose a randomized version of bubble sort that is outlined in Algorithm **??**.

Algorithm 1 Randomized Bubble Sort

```
1: function RandBS($\pi$,$n$)                          ▷ $\pi$ is the permutation of degree $n$ to sort
2:     $CC \leftarrow < >$          ▷ $CC$ will be the sequence of simple transpositions that sorts $\pi$
3:     $LST \leftarrow \{i : \pi[i] > \pi[i+1]\}$
4:     while $LST \neq \emptyset$ do
5:         $i \leftarrow$ RemoveRandomElement($LST$)
6:         Swap $\pi[i]$ and $\pi[i+1]$
7:         Append $i$ to $CC$
8:         if $i > 0$ and $i-1 \notin LST$ and $\pi[i-1] > \pi[i]$ then
9:             Add $i-1$ to $LST$
10:        if $i < n-1$ and $i+1 \notin LST$ and $\pi[i+1] > \pi[i+2]$ then
11:            Add $i+1$ to $LST$
12:    end while
13:    return $CC$
14: end function
```

The *RandBS* algorithm sorts the permutation π (and any array of sortable elements) with the optimal number of adjacent swaps. Indeed, at each iteration of the while loop: (1) a simple transposition is applied to π, thus reducing by one the Kendall-τ distance to e (the ordered permutation), (2) LST contains exactly the simple transpositions that "move" π towards e. This allows also to limit the number of iterations to $\binom{n}{2} = O(n^2)$. Then, it is easy to prove that the time complexity of *RandBS* is $O(n^2)$ as the one of its classical counterpart.

Furthermore, it is worthwhile to notice that *RandBS* produces, as a second result, CC, i.e., a minimal-length sequence of simple transpositions that sorts π. By reversing the sequence CC, the compositions chain of simple transpositions of π is obtained. Interestingly, CC equals to a sequence of edges labels obtained by a "never go back" random walk from π towards e in the subgraph of $\mathit{\Gamma}$ composed by the permutations σ such that $d_K(\pi,\sigma) + d_K(\sigma,e) = d_K(\pi,e)$, where $d_K(\cdot,\cdot)$ is the Kendall-τ distance.

Hence, the application of *RandBS* to $\pi_{r_2}^{-1} \circ \pi_{r_1}$ allows to randomly produce one of its decompositions. Then, by truncating it as aforementioned we obtain $F \cdot \left(\pi_{r_2}^{-1} \circ \pi_{r_1}\right)$ and thus we have a procedure to compute the differential mutation of equation (**??**).

3 Differential Evolution for Permutations

The Differential Evolution for the Permutations space (DEP), outlined in Algorithm **??**, directly evolves a population of NP permutations $\pi_1, \dots, \pi_{NP}$. Its main scheme resembles that of the classical DE with the introduction of a restart mechanism and a memetic local search procedure. Moreover, important variations have been made to the population initialization and to the genetic operators of mutation, crossover and selection. All these components are described in the following.

The population is initialized with $NP - 1$ random permutations and the remaining one is obtained by means of the constructive heuristic $LR(n/m)$ [**?**].

For each population individual π_i, a mutant permutation ν_i is generated according to equation (**??**) and using the procedure described in Section **??**. In

Algorithm 2 Differential Evolution for Permutations

```
 1: Initialize Population
 2: while evaluations budget is not exhausted do
 3:     for $i \leftarrow 1$ to $NP$ do
 4:         $\nu_i \leftarrow$ DifferentialMutation($i$)
 5:         $\upsilon_i^{(1)}, \upsilon_i^{(2)} \leftarrow$ Crossover($\pi_i, \nu_i$)
 6:         Evaluate $f(\upsilon_i^{(1)})$ and $f(\upsilon_i^{(2)})$
 7:     for $i \leftarrow 1$ to $NP$ do
 8:         $\pi_i \leftarrow$ Selection($\pi_i, \upsilon_i^{(1)}, \upsilon_i^{(2)}$)
 9:     if restart criterion then
10:         Perform a Baldwinian Local Search on $\pi_{best}$
11:         Restart Population
12: end while
```

order to avoid the setting of the scale factor F, the self-adaptive scheme proposed in jDE [?] has been used for its online adaptation.

The crossover between the population individual π_i and the mutant ν_i is performed according to the two-point crossover version II (TPII) proposed in [?] and used by AGA [?]. Differently from the classical DE crossover, TPII produces two offspring individuals, i.e., $\upsilon_i^{(1)}$ and $\upsilon_i^{(2)}$. Two indices j, k, such that $1 < j < k < n$, are randomly generated. $\upsilon_i^{(1)}[h] = \pi_i[h]$ for $j \leq h \leq k$ and the missing jobs are placed in $\upsilon_i^{(1)}$ using the order of their appearance in ν_i. Finally, $\upsilon_i^{(2)}$ is filled in the same way but by reversing the role of π_i and ν_i.

In order to choose the trial υ_i that will compete with π_i, a preliminary selection between the two offspring individuals is performed according to $\upsilon_i = \operatorname{argmin}\left\{f(\upsilon_i^{(1)}), f(\upsilon_i^{(2)})\right\}$.

The new population individual π_i' is chosen by a "biased" selection between υ_i and π_i performed according to:

$$\pi_i' = \begin{cases} \upsilon_i & \text{if } f(\upsilon_i) < f(\pi_i) \text{ or } r < \max\{0, 0.01 - \varDelta_i\} \\ \pi_i & \text{otherwise} \end{cases} \tag{6}$$

where r is a random number in $[0, 1]$ and $\varDelta_i$ is the relative fitness variation $(f(\upsilon_i) - f(\pi_i))/f(\pi_i)$. Similarly to classical DE selection, υ_i enters the next generation population if it is fitter than π_i. Otherwise, υ_i may be selected with a small probability that linearly shades from 0.01 when $\varDelta_i = 0$ to 0 when $\varDelta_i = 0.01$. This criterion allows: (1) to slow down the population convergence, (2) to reduce the number of restarts, and (3) to mitigate the super-individual effect observed in some preliminary experiments.

Finally, a restart mechanism has been introduced in order to completely avoid the stagnation of the population. When the population fitnesses are the same, the best individual is kept and the other $NP - 1$ permutations are randomly reinitialized. Furthermore, a local search procedure is applied to the best individual using a Baldwinian approach, that is, the result of the local search is collected but does not enter the DEP population. The local search scheme employed is similar to VNS$_4$ [?] without shakes. A greedy local search using the interchange neighborhood is carried out until a local minimum is found. Then, the best neighbor in its insertion neighborhood is chosen and the process is iterated until a local minimum for both neighborhoods is reached. Moreover, it is worth to notice that the interchange local search iterates by randomly scanning the permutation components at every step and selecting the first improvement found.

4 Experiments

The performances of DEP have been evaluated on the well known 120 benchmark problems proposed by Taillard in [?]. For each problem instance 20 runs were made and the results have been compared with those provided in [?] for the four PFSP-TFT state-of-the-art methods: AGA, VNS$_4$, GM-EDA and HGM-EDA. DEP population size has been set to $NP = 100$ after some preliminary experiments and, in order to provide a fair comparison, the same caps of objective function evaluations reported in [?, Table III] have been adopted.

The performance measure employed is the average relative percentage deviation (ARPD):

$$ARPD = \left(\sum_{i=1}^{20} \frac{(Alg_i - Best) \times 100}{Best} \right) / 20 \tag{7}$$

where Alg_i is the final TFT value found by the algorithm in its i^{th} run, and $Best$ is the best known TFT value for the problem instance at hand.

In order to detect the statistical differences between the performances of DEP and each of the other algorithms, as suggested in [?], we applied to every $n \times m$ problem configuration the non-parametric $1 \times N$ Friedman's test and the Finner post-hoc procedure to the average TFT results produced by each algorithm on every instance.

The best TFT values and the ARPDs of each algorithm are reported in Table 1. The TFTs in bold indicates when DEP reaches the best value and the asterisk denotes when it is a new known optimal TFT. Minimal ARPDs are reported in bold.

Furthermore, for each problem configuration the Friedman's average ranking of all the algorithms are provided. Values in bold denote that DEP significantly outperforms the algorithm, while values in italic denote that DEP is significantly outperformed by the algorithm.

Table 1: Experimental Results

Instance	Best	AGA	VNS$_4$	GM-EDA	HGM-EDA	DEP	Instance	Best	AGA	VNS$_4$	GM-EDA	HGM-EDA	DEP
20 × 5	**14033**	**0.00**	**0.00**	0.18	**0.00**	**0.00**	100 × 5	***253605**	0.29	1.25	0.87	0.23	**0.05**
	15151	**0.00**	**0.00**	0.48	**0.00**	**0.00**		***242579**	0.30	1.80	1.08	0.35	**0.05**
	13301	**0.00**	**0.00**	0.50	**0.00**	**0.00**		***238075**	0.22	1.49	0.85	0.26	**0.07**
	15447	**0.00**	**0.00**	0.43	**0.00**	**0.00**		227889	0.17	1.29	0.78	0.20	**0.06**
	13529	**0.00**	**0.00**	0.21	**0.00**	**0.00**		240589	0.21	1.29	0.80	0.23	**0.02**
	13123	**0.00**	**0.00**	0.08	**0.00**	**0.00**		***232689**	0.32	1.52	0.90	0.28	**0.06**
	13548	**0.00**	**0.00**	0.79	**0.00**	**0.00**		240669	**0.15**	1.34	1.00	0.34	0.25
	13948	**0.00**	**0.00**	0.18	**0.00**	**0.00**		***231064**	0.29	1.79	1.06	0.35	**0.07**
	14295	**0.00**	**0.00**	0.18	**0.00**	**0.00**		***248039**	0.40	1.66	1.05	0.38	**0.09**
	12943	**0.00**	**0.00**	0.46	**0.00**	**0.00**		***243258**	0.19	1.44	1.00	0.28	**0.07**
Avg Rank		2.5	2.5	**5**	2.5	2.5	Avg Rank		2.2	**5**	**4**	**2.7**	1.1
20 × 10	**20911**	**0.00**	**0.00**	0.45	**0.00**	**0.00**	100 × 10	***299101**	0.43	1.63	1.80	0.44	**0.16**
	22440	**0.00**	**0.00**	0.54	**0.00**	**0.00**		***274566**	0.60	1.58	2.08	0.69	**0.28**
	19833	**0.00**	**0.00**	0.31	**0.00**	**0.00**		***288543**	0.37	1.57	1.74	0.38	**0.18**
	18710	**0.00**	**0.00**	0.75	**0.00**	**0.00**		***301552**	0.50	1.79	2.08	0.53	**0.18**
	18641	**0.00**	**0.00**	0.35	**0.00**	**0.00**		***284722**	0.61	1.64	1.95	0.54	**0.22**
	19245	**0.00**	**0.00**	0.77	**0.00**	**0.00**		***270483**	0.42	1.76	1.83	0.45	**0.19**
	18363	**0.00**	**0.00**	0.47	**0.00**	**0.00**		***280257**	0.37	1.58	1.65	0.40	**0.25**
	20241	**0.00**	**0.00**	0.47	**0.00**	**0.00**		***291231**	0.49	1.77	2.03	0.61	**0.27**
	20330	**0.00**	**0.00**	0.27	**0.00**	**0.00**		302624	0.36	1.46	1.76	0.41	**0.20**
	21320	**0.00**	**0.00**	0.24	**0.00**	**0.00**		***291705**	0.48	1.84	1.68	0.50	**0.06**
Avg Rank		2.5	2.5	**5**	2.5	2.5	Avg Rank		2.1	**4.1**	**4.9**	**2.9**	1
20 × 20	**33623**	**0.00**	**0.00**	0.65	**0.00**	**0.00**	100 × 20	***366438**	0.80	1.70	2.26	0.67	**0.37**
	31587	**0.00**	**0.00**	0.28	**0.00**	**0.00**		***373138**	0.55	1.43	2.04	0.58	**0.25**
	33920	**0.00**	**0.00**	0.04	**0.00**	**0.00**		371417	0.47	1.31	1.93	0.36	**0.21**
	31661	**0.00**	**0.00**	0.28	**0.00**	**0.00**		***373574**	0.60	1.36	1.92	0.45	**0.26**
	34557	**0.00**	**0.00**	0.26	**0.00**	**0.00**		***369903**	0.57	1.35	1.92	0.47	**0.19**
	32564	**0.00**	**0.00**	0.30	**0.00**	**0.00**		***372752**	0.51	1.46	2.17	0.42	**0.30**
	32922	**0.00**	**0.00**	0.61	**0.00**	**0.00**		***373447**	0.70	1.82	2.19	0.63	**0.33**
	32412	**0.00**	**0.00**	0.52	**0.00**	**0.00**		385456	0.46	1.41	1.96	0.43	**0.20**
	33600	**0.00**	**0.00**	0.56	**0.00**	**0.00**		***375352**	0.62	1.52	2.01	0.52	**0.41**
	32262	**0.00**	**0.00**	0.41	**0.00**	**0.00**		379899	0.48	1.29	2.05	0.49	**0.46**
Avg Rank		2.5	2.5	**5**	2.5	2.5	Avg Rank		**2.8**	**4**	**5**	2.2	1
50 × 5	64803	**0.05**	0.78	0.79	0.12	**0.05**	200 × 10	1047662	0.48	1.25	1.19	**0.17**	0.21
	68062	**0.06**	0.88	0.94	0.12	0.08		***1035783**	0.94	1.54	1.49	0.32	**0.15**
	63162	**0.19**	1.21	1.34	0.38	0.21		***1045706**	0.66	1.62	1.30	0.32	**0.15**
	68226	0.17	1.12	1.27	0.22	**0.13**		***1029580**	0.77	1.65	1.38	0.45	**0.12**
	69392	**0.09**	0.87	0.89	0.15	**0.09**		***1036464**	0.68	1.35	1.37	0.19	**0.13**
	66841	0.10	0.80	0.82	0.18	**0.04**		1006650	0.50	1.36	1.39	**0.19**	0.23
	66258	0.03	0.74	0.95	0.07	**0.02**		***1052786**	0.95	1.66	1.23	0.24	**0.10**
	64359	**0.05**	0.89	0.97	0.23	**0.05**		***1044961**	0.62	1.51	1.39	0.25	**0.11**
	62981	0.09	0.83	0.81	0.14	**0.05**		***1023315**	0.81	1.61	1.29	0.28	**0.24**
	***68843**	0.15	1.13	1.01	0.29	**0.10**		***1029198**	0.97	1.87	1.48	0.39	**0.25**
Avg Rank		1.6	**4.2**	**4.8**	**3**	1.4	Avg Rank		**3**	**4.8**	**4.2**	1.8	1.2
50 × 10	***87204**	0.33	1.12	2.11	0.39	**0.18**	200 × 20	***1225817**	0.72	1.44	1.68	0.34	**0.16**
	82820	**0.22**	1.09	2.45	0.60	0.30		***1239246**	1.07	1.67	1.66	0.54	**0.21**
	79987	0.23	1.07	1.84	0.36	**0.22**		***1263134**	1.08	1.65	1.57	0.48	**0.26**
	***86545**	0.21	0.94	1.87	0.36	**0.16**		***1233443**	1.25	1.84	1.73	0.58	**0.24**
	86450	**0.14**	0.90	2.02	0.38	0.25		***1220117**	1.12	1.79	1.93	0.53	**0.17**
	86637	0.13	0.77	1.55	0.29	**0.11**		***1223238**	1.17	1.69	1.69	0.46	**0.19**
	88866	**0.25**	0.89	1.97	0.48	0.42		***1237116**	1.03	1.65	1.66	0.64	**0.15**
	***86820**	0.19	0.95	2.04	0.36	**0.01**		***1238975**	1.25	1.72	1.72	0.51	**0.19**
	85526	0.29	1.11	2.10	0.42	**0.28**		***1225186**	1.44	1.91	1.80	0.59	**0.14**
	88077	**0.09**	0.76	2.00	0.45	0.42		***1244200**	1.16	1.62	1.68	0.52	**0.11**
Avg Rank		1.6	**4**	**5**	**3**	1.4	Avg Rank		**3**	**4.5**	**4.5**	2	1
50 × 20	125831	**0.10**	0.65	1.76	0.39	0.14	500 × 20	6708053	**0.11**	0.35	8.90	2.02	1.00
	119259	**0.04**	0.51	1.58	0.22	0.06		6829668	**0.25**	0.38	8.58	1.94	0.66
	116459	**0.19**	0.73	2.24	0.44	0.28		6747387	**0.24**	0.41	8.46	2.04	1.07
	120712	**0.22**	0.61	1.92	0.34	0.34		6787054	**0.26**	0.45	8.75	1.89	0.84
	118184	0.40	0.86	2.30	0.52	**0.39**		6755257	**0.39**	0.41	8.72	1.92	0.74
	120703	0.19	0.62	1.78	0.35	**0.16**		6751496	**0.19**	0.42	8.58	2.13	0.32
	122962	0.38	0.71	2.10	0.47	**0.36**		6708860	**0.27**	0.45	9.15	2.05	0.93
	122489	0.16	0.75	2.24	0.55	**0.14**		6769821	**0.31**	0.58	8.62	2.09	0.73
	121872	0.16	0.76	1.79	0.37	**0.12**		6720474	**0.15**	0.46	8.69	1.91	0.96
	124064	**0.23**	0.90	1.95	0.42	0.29		6767645	**0.19**	0.44	8.51	2.00	0.86
Avg Rank		1.5	**4**	**5**	**3**	1.5	Avg Rank		*1*	*2.1*	**5**	**4**	2.9

In 79 instances over 120, DEP reaches the best TFT, and, most remarkably, in 45 cases they are the new known best values. Moreover, it is worth to notice that DEP has obtained new optima for 23 over 30 instances of size $100 \times m$ and for 18 over 20 instances of size $200 \times m$, which are reputed to be difficult.

The robustness of DEP is proved by the fact that it presents the lowest ARPD results in 96 instances. Again, in almost all 100 and 200 jobs problems, DEP is the best algorithm in average.

Except the case of 500 jobs, DEP has always the lowest Friedman's average rank. The results can be summarized as follows:

– For problems with 20 jobs, all the algorithms perform the same, except GM-EDA which is significantly worse.
– For problems with 50 jobs, DEP has the lowest average rank values and is significantly better than VNS_4, GM-EDA and HGM-EDA.
– For problems with $n = 100$, DEP has the average rank values very close to 1 and has no clear competitor.
– A similar behaviour is found for problems with 200 jobs, but HGM-EDA, although having a worse average rank and obtaining only two best values over 20, is not significantly worse than DEP.
– The only weakness for DEP is found in problems with 500 jobs, where it is outperformed by AGA and VNS_4. Indeed, we observed that the number of restarts was very small or even zero, thus indicating a low convergence rate probably due to the large diameter of the search space.

The conclusion of this analysis is that DEP can be considered among the state-of-the-art PFSP-TFT algorithms and is the best one on the majority of the benchmark problems.

5 Conclusions and Future Works

In this work, a new discrete Differential Evolution algorithm for Permutation spaces (DEP) has been proposed. The main contribution is the differential mutation operator which is defined by means of a randomized bubble sort algorithm and extends the "contour matching" property of classical DE to the permutations space. Moreover, a randomly biased selection operator that allows to improve the population diversity in order to mitigate the super-individual effect has been proposed.

The experimental results on PFSP-TFT show that DEP outperforms the other state-of-the-art algorithms and found 45 new optimal solutions previously unknown.

Promising lines of research for further improvements will focus on the analysis of the contributions of each single DEP component (mutation, crossover, selection, restart, local search) and the tuning of their parameters.

Furthermore, we are planning to investigate the application of the DEP algorithm to other permutation-based problems (like TSP, QAP, LOP, etc.).

Acknowledgments This work was partially supported by Italian Ministry of Education, University and Research (MIUR) under the PRIN 2010-11 grant no. 2010FP79LR_003 "Logical methods of information management", by the University of Perugia, DMI Project "Mobile Knowledge Agents in Evolutionary Environments" and by the services provided by the European Grid Infrastructure (EGI), the Italian Grid Infrastructure (IGI) and the National Grid Initiatives for the Virtual Organization (VO) COMPCHEM.

References

1. Gupta, J., Stafford, J.E. Flowshop scheduling research after five decades. European Journal of Operational Research 169, 699–711 (2006)
2. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A. A Distance-based Ranking Model Estimation of Distribution Algorithm for the Flowshop Scheduling Problem. IEEE Transactions on Evolutionary Computation 99, 1–16 (2013)
3. Costa, W.E., Goldbarg, M.C., Goldbarg, E.G. New VNS heuristic for total flowtime flowshop scheduling problem. Expert Systems with Appl. 39, 8149–8161 (2012)
4. Xu, X., Xu, Z., Gu, X. An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization. Expert Systems with Appl. 38, 7970–7979 (2011)
5. Liu, J., Reeves, C.R. Constructive and composite heuristic solutions to the $P//\sum C_i$ scheduling problem. European Journal of Operational Research 132, 439–452 (2001)
6. Storn, R., Price, K. Differential Evolution:A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Jour.of Global Opt.11, 341–359 (1997)
7. Murata, T., Ishibuchi, H., Tanaka, H. Genetic algorithms for flowshop scheduling problems. Computers & Ind. Eng. 30 (4), 1061–1071 (1996)
8. Milani, A., Santucci, V. Community of scientist optimization: An autonomy oriented approach to distributed optimization. AI Commununications 25, 157-17 (2012)
9. Baioletti, M., Milani, A., Poggioni, V., Rossi, F. Experimental evaluation of pheromone models in ACOPlan. Ann. Math. Artif. Intell. 62(3-4), 187-217 (2011)
10. Onwubolu, G.C., Davendra, D. Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization. Springer-Verlag, Berlin (2009)
11. Cickova, Z., Stevo, S. Flow Shop Scheduling using Differential Evolution. Management Information Systems 5 (2), 8–13 (2010)
12. Li, X., Yin, M. An opposition-based differential evolution algorithm for permutation flowshop scheduling based on diversity measure. Adv.Eng.Soft. 55, 10–31(2013)
13. Price, K.V., Storn, R.M., Lampinen, J.A. Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin (2005)
14. Moraglio, A., Poli, R. Geometric crossover for the permutation representation. Intelligenza Artificiale 5 (1), 49–63 (2011)
15. Schiavinotto, T., Stutzle, T. A review of metrics on permutations for search landscape analysis. Computers & Oper. Res. 34 (10), 3143–3153 (2007)
16. Brest, J., Boskovic, B., Mernik, M., Zumer, V. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Trans. on Evol. Comp. 10 (6), 646–657 (2006)
17. Taillard, E. Benchmarks for basic scheduling problems. European Jour. of Oper. Res. 64 (2), 278–285 (1993)
18. Derrac, J., Garcia, S., Molina, D., Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1, 3–18 (2011)