Solving Permutation Flowshop Scheduling Problems with a Discrete Differential Evolution Algorithm

Valentino Santucci [a]
Marco Baioletti [a]
Alfredo Milani [a,b]
[a] *Dept. of Mathematics and Computer Science*
University of Perugia
Perugia (Italy)
[b] *Department of Computer Science*
Hong Kong Baptist University
Hong Kong (China)
E-mail:
{*milani,baioletti,valentino.santucci*}*@dmi.unipg.it*

In this paper a new discrete Differential Evolution algorithm for the Permutation Flowshop Scheduling Problem with the total flowtime and makespan criteria 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 benchmarks suite and they show that our proposal reaches very good performances compared to other state-of-the-art algorithms. The results are particularly satisfactory on the total flowtime criterion where also new known optima have been found.

Keywords: Permutation Flowshop Scheduling Problem, Differential Evolution, Permutation-based Optimization

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 [1]. In the PFSP there are n jobs $J_1, \ldots, J_n$, each of them composed by a sequence of m operations. O_{ij} is the i-th operation of job J_j and p_{ij} is its processing time. There are m machines $M_1, \ldots, M_m$ and a generic operation O_{ij} can be executed only by machine M_i. Moreover, the execution of any operation cannot be interrupted (no pre-emption) and job passing is not allowed, i.e. the jobs have to be executed following the same order in every machine.

Hence, the goal of PFSP is to determine the best permutation $\pi = \langle \pi[1], \ldots, \pi[n] \rangle$ of the n jobs according to some objective function.

Two important criteria are to minimize the total flowtime (TFT)

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

and the makespan

$$f_{max}(\pi) = \max_{j=1}^{n} c(m, \pi[j]) \tag{2}$$

where, in both equations, $c(i, \pi[j])$ is the completion time of job $\pi[j]$ on machine i and is recursively calculated as

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

in the general case when $i, j > 1$; while in terminal cases:

$$\begin{array}{ll} c(i, \pi[j]) = p_{i,\pi[j]} & \text{if } i = j = 1 \\ c(i, \pi[j]) = p_{i,\pi[j]} + c(i, \pi[j-1]) & \text{if } i = 1 \text{ and } j > 1 \\ c(i, \pi[j]) = p_{i,\pi[j]} + c(i-1, \pi[j]) & \text{if } i > 1 \text{ and } j = 1. \end{array}$$

The PFSP with the TFT criterion has been demonstrated to be NP hard for two or more machines [2], while the makespan criterion is slightly easier, because can be solved in polynomial time for two machines, but remains NP hard for $m > 2$. Therefore, even due to their practical interest, many researches have been devoted to finding high quality and near optimal solutions by means of heuristic or meta-heuristic approaches [3,5,4].

A large number of methods for the PFSP-Makespan have been described and compared in [4] where it is shown that the heuristic NEH [7] is very effective and ILS was the best meta-heuristic algorithm [27]. More recently an iterated greedy (IG) algorithm has been introduced in [6] and it can be considered as the state-of-the-art for this problem criterion. IG adopts a simulated annealing like acceptance criterion and iteratively alternates a destruction-construction procedure and a local search scheme. The initial solution is build with the NEH heuristic.

A report of the state-of-the-art methods for PFSP-TFT has been recently provided in [5] where it is shown that the most performing meta-heuristics are: VNS$_4$ [10], AGA [11] and GM-EDA together with its hybrid variant HGM-EDA [5]. VNS$_4$ applies a variable neighborhood search (VNS) to an initial permutation built by means of a constructive heuristic called LR(n/m) [12]. 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 [13] is a popular evolutionary algorithm (see for example [14,15]) for continuous optimization problems. 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 [16] or the more recents [17,18]) adopt some transformation scheme to encode permutations into numerical vectors. This distinction between the phenotypic and genotypic space [20] 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 problem, using both criteria, that works directly on the permutations space. Since the differential mutation operator has been generally considered the key component of DE [19], 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 2. The full DE scheme for PFSP is described in Section 3. An experimental analysis of the proposed approach for both the criteria is provided in Section 4. Finally, conclusions are drawn in Section 5 where some future lines of research are also depicted.

2. Differential Mutation in the Permutations Space

Differential Evolution (DE) [13] is a popular and powerful evolutionary algorithm over continuous search spaces using the differential mutation operator as its key component [19]. 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 [19]), 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 $\pi^{-1} \in S(n)$ and the ordered permutation e is the neutral element.

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 (3) 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 (4) 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 $S(n)$ [22]:

– the set of all transpositions $T = \{(i,j)_T : 1 \leq i < j \leq n\}$, where $(i,j)_T$ denotes the permutation which only swaps the elements at places i and j,
– the set of all insertions $I = \{(i,j)_I : i \neq j \text{ and } 1 \leq i,j \leq n\}$, where $(i,j)_I$ denotes the permutation that shifts the element at place j to place i,
– the set of all simple transpositions $ST = \{(i,i{+}1)_T : 1 \leq i \leq n-1\}$, 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 [22]. 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 d_K [22] and equals the number of inversions of either $\pi_2^{-1} \circ \pi_1$ or $\pi_1^{-1} \circ \pi_2$.

Here, we have decided to focus on ST. The choice among the generating sets has been influenced by the observation that a smaller number of generators induces a higher search space diameter and generally longer paths connecting two solutions. This looks to be more suited for a differential mutation operator. Indeed, a shortest path in the ST space is generally longer than in the two other spaces and is composed by "many little steps", each of them more likely to produce slight variations in the objective function. The truncation operator, if performed on spaces with smaller diameter and shorter paths, like T and I, would have a reduced number of possible values and we expect that it would produce worst results. Note also that ST induces a smaller "branching factor" on the search space, thus reducing the arbitrariness of the navigation. These considerations can be in general extended to other combinatorial optimizazion 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 1.

The *RandBS* algorithm sorts the permutation π (and any array of comparable elements) with the optimal number $d_K(\pi, e)$ 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
2. LST contains exactly the simple transpositions that move π towards e.

Algorithm 1 Randomized Bubble Sort

```
1: procedure RandBS($\pi$,$n$)
2:   $CC \leftarrow <>$
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 $(i, i+1)_T$ 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:    end if
11:    if $i < n-1$ and $i+1 \notin LST$ and $\pi[i+1] > \pi[i+2]$ then
12:      Add $i+1$ to $LST$
13:    end if
14:  end while
15:  return $CC$
16: end procedure
```

. Since the number of iterations is $\binom{n}{2} = O(n^2)$, as the diameter of ST, and all the operations inside the loop can be implemented in $O(1)$, 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. CC equals to a sequence of edges labels obtained by a "never go back" random walk from π towards e in the subgraph of Γ composed by the permutations σ such that $d_K(\pi, \sigma) + d_K(\sigma, e) = d_K(\pi, e)$.

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 $F \cdot (\pi_{r_2}^{-1} \circ \pi_{r_1})$ and thus we have a procedure to compute the differential mutation of equation (5).

3. Differential Evolution for Permutations

The Differential Evolution for the Permutations space (DEP), outlined in Algorithm 2, directly evolves a population of NP permutations $\pi_1, \ldots, \pi_{NP}$. 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 subsections. Note that, both in Algorithm 2 and in the following descriptions, depending on the chosen objective function, f can interchangeably refer to f_{sum} or f_{max}.

Algorithm 2 DE 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 $f\left(\upsilon_i^{(1)}\right)$ and $f\left(\upsilon_i^{(2)}\right)$
7:   end for
8:   for $i \leftarrow 1$ to $NP$ do
9:     $\pi_i \leftarrow$ Selection $\left(\pi_i, \upsilon_i^{(1)}, \upsilon_i^{(2)}\right)$
10:  end for
11:  if restart criterion then
12:    Perform a Local Search on $\pi_{best}$
13:    Restart Population
14:  end if
15: end while
```

3.1. Initialization

The population is initialized with $NP - 1$ random permutations (obtained by means of the uniformly random permutation generator known as Fisher-Yates shuffle [23]) and the remaining permutation is built using the appropriate constructive heuristic for the PFSP objective at hand, i.e., $LR(n/m)$ [12] for the total flowtime and NEH [7] for the makespan.

3.2. Differential Mutation

For each population individual π_i, three parents $\pi_{r_0}, \pi_{r_1}, \pi_{r_2}$ (different among them and with respect to π_i) are randomly selected from the current population and a mutant permutation ν_i is generated according to equation (5) and using the procedure described in Section 2.

Furthermore, in order to avoid the setting of the scale factor F, the popular self-adaptive scheme proposed in jDE [24] has been used for its adaptation during the evolution. Basically, it introduces a self-adapting parameter F_i for each individual. Just before the mutation a temporary F_{mutant} is generated according to

$$F_{mutant} \leftarrow \begin{cases} 0.1 + r_1 \cdot 0.9 & \text{if } r_2 < 0.1 \\ F_i & \text{otherwise} \end{cases}$$

where r_1, r_2 are two random number in $[0, 1]$. The mutation is performed by using F_{mutant}, and, in the case that one of the offsprings replaces the original population individual, also F_{mutant} replaces F_i.

3.3. Crossover

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 [25] and used by AGA [11]. 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 $j \le h \le k$ 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.

3.4. Selection

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:

$$\upsilon_i = \text{argmin}\left\{f\left(\upsilon_i^{(1)}\right), f\left(\upsilon_i^{(2)}\right)\right\} \tag{6}$$

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

$$\pi_i' = \begin{cases} \upsilon_i & \text{if } f(\upsilon_i) < f(\pi_i) \text{ or } r < \max\{0, \alpha - \Delta_i\} \\ \pi_i & \text{otherwise} \end{cases} \tag{7}$$

where r is a random number in $[0,1]$, Δ_i is the relative fitness variation $\dfrac{f(\upsilon_i) - f(\pi_i)}{f(\pi_i)}$, and $\alpha \in [0,1]$ is a selection parameter.

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 α when $\Delta_i = 0$ to 0 when $\Delta_i = \alpha$. It is worthwhile to note that, when $\alpha = 0$, the classical DE selection scheme is reproduced.

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.

3.5. Restart

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.

3.6. Local Search

A local search procedure is performed at every restart and it is applied to the best individual.

The local search scheme employed is similar to VNS$_4$ [10] 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 by selecting the first improving neighbor found.

4. Experiments

Both for TFT and makespan criteria, the performances of DEP have been evaluated on the well known 120 benchmark problems proposed by Taillard in [26]. 20 runs for each problem instance were made and the results have been compared with the state-of-the-art algorithms for both the criteria.

The termination criterion for DEP and the competitor algorithms has been set according to [5]. In particular, the budgets of objective function evaluations reported in [5, Table III] have been adopted.

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

$$ARPD = \frac{1}{20}\sum_{i=1}^{20} \frac{(Alg_i - Best) \times 100}{Best} \tag{8}$$

where Alg_i is the final objective value found by the algorithm in its i^{th} run, and $Best$ is the best known objective 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 [28], we applied to every $n \times m$ problem configuration the non-parametric Friedman's test and the Finner post-hoc procedure to the average results produced by each algorithm on every instance. 0.05 has been adopted as confidence level.

4.1. Total Flowtime Criterion

For the TFT criterion, preliminary experiments have been held in order to calibrate the DEP parameters. The population size NP has been set to 100, the selection parameter to $\alpha = 0.01$ and the local search has been applied using the Baldwinian style, i.e., the improved solution is recorded but does not enter DEP population.

DEP performances have been compared with those provided in [5] for the four PFSP-TFT state-of-the-art methods: AGA [11], VNS_4 [10], GM-EDA and HGM-EDA [5].

The best TFT values and the ARPDs of each algorithm are reported in Table 1. The TFTs in bold indicate when DEP reaches the best value, while 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 ranks 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.

The results show that, 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 optimal values 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. This may suggest that the best values obtained are probably the optimal values for these instances.
– 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 100 jobs, DEP has the average rank values very close to 1 and there is no clear competitor with DEP.
– 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. This is probably due to a very slow convergence of the DEP population caused by the extremely large diameter of the search space where the differential mutation navigates (see Section 2). This thesis is also corroborated by the observation that very few or even zero restart operations were performed.

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.

4.2. Makespan Criterion

Also for the makespan criterion, preliminary experiments have been held in order to calibrate the DEP parameters. The population size NP has been set to 20, the selection parameter to $\alpha = 0.01$ and the local search has been applied using the Lamarckian style, i.e., the improved solution enters DEP population.

DEP performances have been compared with those of IG, i.e., the iterated greedy algorithm described in [6], GM-EDA and HGM-EDA [5].

The best makespan values and the ARPDs of each algorithm are reported in Table 2. Makespan values in bold indicates when DEP reaches the best value. Also minimal ARPDs are reported in bold. Furthermore, for each problem configuration, the Friedman's average ranks 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 2 show that DEP reaches the best makespan in 85 instances over 120. Moreover, the robustness of DEP is even better, since it presents the lowest ARPDs in 91 instances.

Differently from TFT, the results are here discussed by aggregating on the number of machines:

- For the problems with 5 machines, DEP has one of the lowest Friedman's average rank and its performances are significantly better than those of GM-EDA, while are comparable with respect to IG and HGM-EDA.
- For the problems with 10 machines, DEP performances, although very good on 20×10 and 50×10 instances, looks to degrade increasing the number of jobs, at least with respect to HGM-EDA.
- The best performances of DEP are found on the 20 machines problems. Here, DEP has always the lowest Friedman's average rank and, most remarkably, it reaches the best ARPD value on all the 30 instances with 100, 200, and 500 jobs.

Summarizing, the performances of DEP seems to increase with the number of machines. Since, for the makespan criterion, the neutrality[1] of the problem landscape plausibly decreases with the number of machines, the results probably indicates that DEP performs well when the neutrality is not too high.

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.

Experiments were held on the permutation flowshop scheduling problem (PFSP), both for the total flowtime (TFT) and makespan criteria. 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. Also the experiments on PFSP-Makespan show very good results for DEP and give the indication that DEP performances degrade when the landscape neutrality increases.

[1]In problem landscapes, the neutrality property refers to the amount of equal-quality neighbors (see for example [29, Ch. 5]).

Promising lines of research for further improvements will focus on the the application of the DEP algorithm to other permutation-based problems (like TSP, QAP, LOP, etc.) and a deeper investigation of DEP performances in relation with the problem landscape properties, Finally we are also planning to implement the discrete differential mutation using generic transpositions and insertions as generating sets.

References

[1] Gupta, J., Stafford, J.E.: Flowshop scheduling research after five decades. European Journal of Operational Research 169, 699–711 (2006)

[2] Gonzalez, T., Sahni, S.: Flowshop and Jobshop Schedules: Complexity and Approximation. Operations Research 26 (1), 36–52 (1978)

[3] Pan, Q.K., Ruiz, R.: A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. Computers and Industrial Engineering 55 (4), 795–816 (2013)

[4] Ruiz, R., Maroto, C.: A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research 165, 479–494 (2005)

[5] 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)

[6] Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation Flowshop Scheduling Problem. European Journal of Operational Research 177, 2033-2049 (2007)

[7] Nawaz, M., Enscore Jr., E.E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA, The International Journal of Management Science 11 (1), 91-95 (1983)

[8] Cheng, V.C., Leung, C.H.C., Liu, J., Milani, A. Probabilistic Aspect Mining Model for Drug Reviews. IEEE Transactions on Knowledge and Data Engineering 99, p.1 (2014)

[9] Franzoni, V., Milani, A. Heuristic semantic walk for concept chaining in collaborative networks. International Journal of Web Information Systems 10 (1), 85–103 (2014)

[10] 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)

[11] 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)

[12] 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)

[13] 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)

[14] Milani, A., Santucci, V. Community of scientist optimization: An autonomy oriented approach to distributed optimization. AI Commununications 25, 157-17 (2012)

[15] 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)

[16] Onwubolu, G.C., Davendra, D. Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization. Springer-Verlag, Berlin (2009)

[17] Cickova, Z., Stevo, S. Flow Shop Scheduling using Differential Evolution. Management Information Systems 5 (2), 8–13 (2010)

[18] Li, X., Yin, M. An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Adv. in Eng. Soft. 55, 10–31 (2013)

[19] Price, K.V., Storn, R.M., Lampinen, J.A. Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin (2005)

[20] Rothlauf, F. Representations for Genetic and Evolutionary Algorithms. Springer, Berlin (2006)

[21] Moraglio, A., Poli, R.: Geometric crossover for the permutation representation. Intelligenza Artificiale 5 (1), 49–63 (2011)

[22] Schiavinotto, T., Stutzle, T. A review of metrics on permutations for search landscape analysis. Computers & Oper. Res. 34 (10), 3143–3153 (2007)

[23] Fisher, S., Yates, F. Statistical tables. Oliver & Boyd, Edinburgh (1943)

[24] 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)

[25] Murata, T., Ishibuchi, H., Tanaka, H. Genetic algorithms for flowshop scheduling problems. Computers & Ind. Eng. 30 (4), 1061–1071 (1996)

[26] Taillard, E. Benchmarks for basic scheduling problems. European Jour. of Oper. Res. 64 (2), 278–285 (1993)

[27] Stützle, T.: Applying iterated local search to the permutation flow-shop problem. Tech. Rep. AIDA-98-04, FG Intellektik, TU Darmstadt (1998)

[28] 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)

[29] H. H. Hoos and T. Stutzle, "Stochastic Local Search: Foundations and Applications," Morgan Kaufmann Publishers, San Francisco, CA, USA, 2005.

Acknowledgments

We gratefully acknowledge funding from the Italian Ministry of Education, University and Research (MIUR) through PRIN project "Mobile Knowledge Agents in Evilutionary Environments". Moreover, the research leading to the results presented in this paper has been possible thanks to the grid resources and services provided by the European Grid Infrastructure (EGI), the Italian Grid Infrastructure (IGI) and the National Grid Initiatives that support the Virtual Organization (VO) COMPCHEM.

Table 1
Experimental Results for Total Flowtime Criterion

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

Table 2
Experimental Results for Makespan Criterion

Instance	Best	IG	GM-EDA	HGM-EDA	DEP	Instance	Best	IG	GM-EDA	HGM-EDA	DEP
20 × 5	**1278**	**0.00**	**0.00**	**0.00**	**0.00**	100 × 5	**5493**	**0.00**	0.04	**0.00**	**0.00**
	1359	**0.00**	0.07	**0.00**	**0.00**		5268	0.13	0.32	**0.00**	0.13
	1081	**0.00**	0.37	**0.00**	**0.00**		**5175**	**0.00**	0.25	**0.00**	**0.00**
	1293	**0.00**	0.15	**0.00**	**0.00**		5014	0.08	0.24	**0.00**	0.06
	1235	**0.00**	0.08	**0.00**	**0.00**		**5250**	**0.00**	0.10	**0.00**	**0.00**
	1195	**0.00**	1.09	**0.00**	**0.00**		**5135**	**0.00**	0.19	**0.00**	**0.00**
	1234	0.31	1.38	**0.24**	0.41		**5246**	**0.00**	0.23	**0.00**	**0.00**
	1206	**0.00**	0.17	**0.00**	**0.00**		**5094**	0.01	0.22	**0.00**	**0.00**
	1230	**0.00**	0.57	**0.00**	**0.00**		**5448**	**0.00**	0.46	**0.00**	**0.00**
	1108	**0.00**	**0.00**	**0.00**	**0.00**		5322	0.06	0.30	**0.00**	0.04
Avg Rank		2.1	**3.7**	2	2.2	Avg Rank		**2.45**	**4**	1.6	1.95
20 × 10	**1582**	**0.00**	0.51	**0.00**	**0.00**	100 × 10	5770	0.19	0.78	**0.00**	0.10
	1659	**0.00**	0.66	**0.00**	**0.00**		5349	0.30	0.71	**0.06**	0.24
	1496	**0.00**	1.14	**0.00**	**0.00**		5676	0.22	0.07	**0.04**	0.05
	1377	**0.00**	1.02	**0.00**	**0.00**		5781	1.05	1.16	**0.24**	0.66
	1419	**0.00**	0.49	**0.00**	**0.00**		5467	0.91	0.77	**0.29**	0.52
	1397	**0.00**	0.50	**0.00**	**0.00**		**5308**	0.12	0.32	**0.00**	**0.00**
	1484	**0.00**	0.27	**0.00**	**0.00**		5596	0.30	0.79	**0.04**	0.12
	1538	0.26	0.91	0.07	**0.00**		5623	0.54	0.92	**0.23**	0.48
	1593	**0.00**	1.13	**0.00**	**0.00**		5875	0.74	0.87	**0.00**	0.48
	1591	0.09	1.01	**0.00**	**0.00**		5848	0.91	0.75	**0.00**	0.54
Avg Rank		2.2	**4**	1.95	1.85	Avg Rank		**3.3**	**3.7**	*1.05*	1.95
20 × 20	**2297**	0.02	18.94	**0.00**	**0.00**	100 × 20	**6245**	2.18	2.50	0.58	**0.06**
	2099	**0.00**	35.59	**0.00**	**0.00**		**6210**	2.34	1.84	0.66	**0.13**
	2326	0.09	12.98	**0.00**	**0.00**		**6303**	2.04	1.87	0.44	**0.07**
	2223	**0.00**	24.52	**0.00**	**0.00**		6291	1.60	1.54	0.48	**0.23**
	2291	0.12	25.01	0.04	**0.00**		**6362**	1.82	2.12	0.36	**0.05**
	2226	0.09	27.27	**0.00**	**0.00**		**6423**	1.85	1.59	0.48	**0.02**
	2273	**0.00**	20.37	**0.00**	**0.00**		**6298**	2.39	1.73	0.54	**0.10**
	2200	0.15	22.82	**0.00**	**0.00**		**6423**	2.76	2.52	0.97	**0.15**
	2237	**0.00**	14.48	**0.00**	**0.00**		**6292**	2.52	2.10	0.81	**0.17**
	2178	0.08	27.73	**0.00**	**0.00**		**6476**	1.47	1.64	0.36	**0.06**
Avg Rank		**2.6**	**4**	1.75	1.65	Avg Rank		**3.7**	**3.3**	**2**	1
50 × 5	**2724**	**0.00**	0.29	**0.00**	**0.00**	200 × 10	10872	0.16	0.33	**0.00**	0.31
	2834	0.10	0.42	**0.00**	**0.00**		10493	0.47	0.45	**0.03**	0.79
	2621	**0.00**	0.27	**0.00**	**0.00**		10922	0.38	0.81	**0.00**	0.68
	2751	**0.00**	0.62	**0.00**	**0.00**		10889	1.17	0.26	**0.01**	0.20
	2863	**0.00**	0.03	**0.00**	**0.00**		10527	0.14	0.09	**0.00**	0.15
	2829	**0.00**	0.14	**0.00**	**0.00**		10330	0.35	0.51	**0.01**	0.45
	2725	**0.00**	0.40	**0.00**	**0.00**		10857	0.27	0.52	**0.00**	0.47
	2683	**0.00**	0.71	**0.00**	**0.00**		10731	0.26	0.62	**0.02**	0.61
	2552	0.16	0.35	**0.00**	0.06		10438	0.26	0.37	**0.03**	0.27
	2782	**0.00**	**0.00**	**0.00**	**0.00**		10676	0.49	0.52	**0.02**	0.51
Avg Rank		2.25	**3.85**	1.9	2	Avg Rank		*2.4*	3.5	*1*	3.1
50 × 10	**3025**	0.46	1.22	**0.00**	**0.00**	200 × 20	**11243**	1.98	1.46	0.56	**0.24**
	2877	1.47	1.81	0.45	**0.33**		**11269**	2.71	1.59	0.40	**0.09**
	2852	0.91	1.54	0.42	**0.32**		**11397**	1.72	1.32	0.30	**0.02**
	3063	0.42	0.85	**0.00**	**0.00**		**11345**	2.22	1.33	0.31	**0.04**
	2979	1.31	1.54	**0.57**	0.88		**11293**	1.71	1.37	0.28	**0.10**
	3006	0.91	1.73	0.03	**0.00**		**11234**	1.90	1.43	0.57	**0.11**
	3098	0.88	2.07	**0.26**	0.38		**11424**	1.58	1.16	0.23	**0.04**
	3038	0.34	0.79	**0.10**	0.20		**11402**	2.06	1.11	0.29	**0.13**
	2900	0.59	1.14	0.07	**0.06**		**11241**	2.35	1.61	0.70	**0.23**
	3078	1.66	1.40	0.16	**0.00**		**11339**	2.21	1.19	0.35	**0.04**
Avg Rank		**3.1**	**3.9**	1.6	1.4	Avg Rank		**2**	**4**	**3**	1
50 × 20	3870	1.19	2.14	**0.49**	0.59	500 × 20	**26182**	1.15	3.38	0.23	**0.02**
	3711	1.96	3.18	0.27	**0.08**		**26716**	0.83	2.77	0.18	**0.05**
	3658	2.09	3.03	0.55	**0.10**		**26494**	0.83	3.71	0.31	**0.10**
	3739	1.14	2.09	0.43	**0.12**		26558	1.12	3.19	**0.15**	0.12
	3625	1.75	1.93	**0.39**	0.40		**26379**	0.93	2.30	0.28	**0.04**
	3698	1.47	2.54	0.32	**0.27**		**26581**	0.78	2.99	0.12	**0.01**
	3720	1.60	2.23	0.30	**0.15**		**26424**	0.77	3.17	0.45	**0.21**
	3712	2.11	2.45	0.43	**0.22**		**26646**	0.89	2.64	0.33	**0.00**
	3754	1.45	2.21	0.51	**0.37**		**26123**	1.01	3.08	0.28	**0.11**
	3768	1.65	2.65	0.40	**0.07**		**26551**	0.88	3.14	0.21	**0.06**
Avg Rank		**3**	**4**	**1.8**	1.2	Avg Rank		**3**	**4**	**2**	1