Computers & Operations Research 36 (2009) 2638 – 2646

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems

Bassem Jarboui[a], Mansour Eddaly[a], Patrick Siarry[b,*]

[a]FSEGS, route de l'aéroport km 4, Sfax 3018, Tunisia
[b]LiSSi, Université de Paris 12, 61 avenue du Général de Gaulle, 94010 Créteil, France

ARTICLE INFO

Available online 21 November 2008

Keywords:
Estimation of distribution algorithm
Variable neighbourhood search
Scheduling
Permutation flowshop
Flowtime

ABSTRACT

In this work we propose an estimation of distribution algorithm (EDA) as a new tool aiming at minimizing the total flowtime in permutation flowshop scheduling problems. A variable neighbourhood search is added to the algorithm as an improvement procedure after creating a new offspring. The experiments show that our approach outperforms all existing techniques employed for the problem and can provide new upper bounds.

1. Introduction

In a permutation flowshop scheduling problem (PFSP), there is a set of n jobs that must be processed on a set of m machines in the same order, such that each job is processed on machine 1 in first place, machine 2 in second place,…, and machine m in the last place. Gupta and Stafford [1] have presented a review of developments in flowshop scheduling over the last fifty years. They have listed the assumptions regarding the flowshop scheduling problem in general and the permutation flowshop scheduling problem in particular. The processing times are fixed, known in advance and have non-negative values. All jobs are available for processing at time 0 and all machines are available over the scheduling period. In addition, each machine can process at most one job and each job can be processed on at most one machine. Besides, no pre-emption is allowed, i.e., once the processing of a job on a machine has started, it must be completed without interruption. The most common criteria are the makespan and the total flowtime. This paper addresses the total flowtime (*TFT*) performance measure. This criterion measures the time a job stays in the system, minimizing that time leads to maximizing the utilization of resources. Concerning its complexity, the PFSP has been proved to be NP-complete in the strong sense for more than two machines (Garey et al. [2]).

Diverse approaches were proposed to solve this problem with respect to *TFT* criterion, including exact algorithms such as branch and bound algorithm (Chung et al. [3], Velde [4], Bansal [5], Ignal and Schrage [6]), constructive heuristics (Woo and Yim [7], Framinan and Leisten [8], Liu and Reeves [9], Ho and Gupta [10], Framinan et al. [11], Rajendran [12], Allahverdi and Aldowaisan [13], Li et al. [14]) and metaheuristics like genetic algorithm (GA) (Vempati et al. [15], Zhang et al. [16]), ant colony optimization (ACO) (Gajpal and Rajendran [17], Rajendran and Ziegler [18]) and particle swarm optimization (PSO) (Jarboui et al. [19], Tasgetiren et al. [20]).

Estimation of distribution algorithm (EDA) was introduced by Mühlenbein and Paaß [21]. It constitutes a new tool of evolutionary algorithms (Larranaga and Lozano [22]) based on the probabilistic model learned from a population of individuals. Starting with a population of individuals (candidate solutions), generally generated randomly, this algorithm selects good individuals with respect to their fitness. Then a new distribution of probability is estimated from the selected candidates. Next, new offspring is generated from the estimated distribution. The process is repeated until the termination criterion is met.

In order to improve the quality of EDA solution, it is recommended to use a local search algorithm (Lozano et al. [23]). We propose to use the variable neighbourhood search (VNS) (Mladenović and Hansen [24], Hansen and Mladenović [25]) to improve the performance of our EDA.

This paper is organized as follows: Section 2 presents a formulation of the permutation flowshop scheduling problem. Section 3 describes the basic estimation of distribution algorithm. The EDA for the PFSP is presented in Section 4. Section 5 presents the hybrid EDA. The computational results are presented in Section 6 and a conclusion is given in Section 7.

2. The permutation flowshop scheduling problem (PFSP)

In a PFSP, there is a set of n jobs to be processed through a set of m machines, where each job $j(j = 1, 2, \ldots, n)$ passes through

* Corresponding author.
E-mail addresses: bassem_jarboui@yahoo.fr (B. Jarboui), eddaly.mansour@gmail.com (M. Eddaly), siarry@univ-paris12.fr (P. Siarry).

0305-0548/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2008.11.004

the machines $1, 2, \ldots, m$ in that order, without interruption of the processing (no pre-emption). Referring to the notation of Graham et al. [26] ($\alpha/\beta/\gamma$), the PFSP for *TFT* criterion is denoted by *F/permu/TFT*. Let p_{ij} be the processing time for the job j on the machine $i (i = 1, 2, \ldots, m)$, and C_{ij} denote the completion time of job j on machine i. Then $C_{[j]i}$ is the completion time of the job scheduled in the jth position in the sequence on machine i. $C_{[j]i}$ is computed as: $C_{[j]i} = p_{[j]i} + \max\{C_{[j]i-1}, C_{[j-1]i}\}$. So, we can obtain the *TFT* as follows: $TFT = \sum_{j=1}^{n} C_{[j]m}$.

3. Estimation of distribution algorithm (EDA)

In 1996, a new tool of evolutionary algorithms, called "Estimation of Distribution Algorithm" (Mühlenbein and Paaß [21], Larranaga and Lozano [22]), was proposed. Unlike other approaches of evolutionary algorithms, EDA uses neither crossover nor mutation. Therefore, it generates new offspring according to a probabilistic model learned from a population of parents. The main steps of the canonical EDA are described as follows (Lozano et al. [23]): first, a random initial population is commonly used. Second, a subpopulation of Q parent individuals is selected with a selection method based on the fitness function. Third, the probability of distribution of the selected parents is estimated by a probabilistic model. Fourth, new offspring are generated according to the estimated probability. Finally, some individuals in the current population are replaced with new generated offspring. These steps are repeated until one stopping criterion is met. In the combinatorial context, several EDA applications were developed, such as knapsack problem, travelling salesman problem, clustering and jobshop scheduling problems (Larranaga and Lozano [22]).

4. Our proposed EDA for PFSP

For solving the PFSP with respect to the *TFT* minimization, our proposed solution algorithm is based on the EDA. A new way for the design of the probabilistic model is proposed and the framework of the algorithm is introduced hereafter for discussion.

4.1. Solution representation and initial population

The well-known job based encoding scheme is frequently used in the literature of the PFSP, so it is also used in our algorithm. In this representation, the jth number in the permutation denotes the job located in position j. In order to guarantee the diversification in the population, we use an initial random population of P individuals, uniformly distributed.

4.2. Selection

In our algorithm we adopted the selection procedure of Reeves [27] for solving the flowshop scheduling problem. We describe this procedure as follows:

(i) first, for each individual p, calculate the fitness value $f(p) = 1/TFT(p)$; (ii) second, the individuals of the initial population are sorted in ascending order according to their fitness, i.e. the individual with a higher *TFT* value will be at the top of the list. Finally, the selection of parents is made relatively to the probability:

$$prob(r) = \frac{2r}{P(P+1)}$$

where r is the rank of the rth individual in the sorted list.

4.3. Probabilistic model and creation of new individuals

The probabilistic model constitutes the main issue for EDA and the performance of the algorithm is closely related to it (Lozano et al. [23]), the best choice of the model is crucial. This step consists in building an estimation of distribution for the subset of Q selected individuals. In our algorithm, we determine the estimation of distribution model while taking into account both the order of the job in the sequence and the similar blocks of jobs presented in the selected parents.

Let:

- η_{jk} be the number of times of appearance of job j before or in the position k in the subset of the selected sequences augmented by a given constant δ_1. The value of η_{jk} refers to the importance of the order of the jobs in the sequence.
- $\mu_{j[k-1]}$ be the number of times of appearance of job j immediately after the job in the position $k-1$ in the subset of the selected sequences augmented by a given δ_2. $\mu_{j[k-1]}$ indicates the importance of the similar blocks of jobs in the sequences. In such way, we prefer to conserve the similar blocks as much as possible.

We note that δ_1 and δ_2 are two parameters used for the diversification of the solutions. Indeed, we employed these parameters in order to slow down the convergence of the algorithm.

- Ω_k: the set of jobs not already scheduled until position k.

We define π_{jk} the probability of selection of the job j in the kth position by the following formula: $\pi_{jk} = \eta_{jk} \times \mu_{j[k-1]} / \sum_{l \in \Omega_k} (\eta_{lk} \times \mu_{l[k-1]})$. According to this probability, for each position k, we select a job j from the set of not already scheduled jobs in the sequence of a new individual.

4.4. Replacement

Replacement is the last phase in the EDA, it consists in updating the population. Therefore, at each iteration, O offspring are generated from the subset of the selected parents. There are many techniques for deciding if the new individuals will be added to the population.

In our algorithm, we compare the new individual with the worst individual in the current population. If the offspring is best than this individual and the sequence of the offspring is unique, then the worst individual is removed from the population and will be replaced with the new individual.

4.5. Stopping criterion

The stopping condition indicates when the search will be terminated. Various stopping criteria may be listed, such as maximum number of generations, bound of time, maximum number of iterations without improvement, etc. We set a maximum number of iterations and a maximal computational time in our algorithm.

5. Hybrid EDA for PFSP (EDA–VNS)

Aiming at improving the performance of EDA and preventing it from being stuck into a local optimum, a successful way is to hybridize it with local search methods (Lozano et al. [23]). We propose to apply a VNS algorithm (Mladenović and Hansen [24], Hansen and Mladenović [25]) as an improvement procedure after the creation of a new individual.

```
procedure swap_local_search(x_0)
    x_best = x_0;                                   // x_best is the best solution obtained by swap local search procedure
    set i = 1;
    do
        set j = i + 1;
        while(j ≤ n)
            x' = permute the jobs in the positions i and j in x_best
            if (TFT(x') < TFT(x_best)) then         // check the improvement of best solution
                x_best = x';
                i = i − 1;
                j = i + 1;
            else
                j = j + 1;
            endif
        endwhile
        i = i + 1;
        if(i > n − 1) then  i = 1;
    until (no possible improvement)
    return x_best;
end
```

Fig. 1. Pseudo code of swap local search procedure.

```
procedure insert_local_search(x_0)
    x_best = x_0;                                   // x_best is the best solution obtained by insert local search procedure
    set i = 1;
    do
        set j = 1;
        while(j ≤ n)
            x' = insert the jobs in the position i to position j in x_best
            if (TFT(x') < TFT(x_best)) then         // check the improvement of best solution
                x_best = x';
                i = i − 1;
                j = 1;
            else
                j = j + 1;
            endif
        endwhile
        i = i + 1;
        if(i > n) then  i = 1;
    until (no possible improvement)
    return x_best;
end
```

Fig. 2. Pseudo code of insert local search procedure.

5.1. Variable neigbourhood search algorithm

For the application of VNS algorithm, we propose to restrict it to a subset of the individuals by employing a probability of improvement that depends on the quality of the related individual. We define this probability as follows:

Let $p^c = \max\{\exp(RD/\alpha), \varepsilon\}$ be the calculated probability for application of VNS where $RD = (f(x_{current}) - f(x_{best}))/f(x_{best})$, $x_{current}$ denotes the created offspring and x_{best} denotes the best solution found by the algorithm. For each individual, we draw at random a number between 0 and 1. If this number is less than or equal to p^c, then we apply VNS to the individual under consideration.

We select two structures of neighbourhoods, called *swap_local_search* and *insert_local_search*. The first one leads to all possible swaps of pairs of job's positions (i, l), where $1 \leqslant i < l \leqslant n$, within all parts of solutions (Fig. 1). If the swap moves were performed and a local optimum was found, the second structure of neighbourhood consists of all possible insert moves of pairs of positions of jobs (i, l), within all parts of the so obtained solution (Fig. 2). Then, we return to the *swap_local_search* with the improved solution as a current solution, i.e., the local optimum resulted after the application of the *insert_local_search*. We reapply this procedure until no improvement in the solution.

If the obtained solution is better than the best solution, then we replace the latter with the new solution and we reinitialize the number of iterations at 0, else we increment the number of iterations. Next, we select two distinct positions (i,j) at random following the uniform distribution in the range $[1, n]$ from x_{best}, and the jobs on these positions are exchanged. The whole procedure is repeated as far as reaching the maximal number of iterations ($iter_{\max}$). The pseudo code of the VNS algorithm is given in Fig. 3.

6. Computational results

The algorithms were coded in C++ programming language. All experiments were run on a desktop PC with Intel Pentium IV, Windows XP, 3.2 GHz processor and 1 GB memory. For the evaluation of the results, the benchmark problems from Taillard [28] were considered, with $m = 5, 10$ and 20 and $n = 20, 50$ and 100.

The performance measure employed in our numerical study was $\Delta_{average}$, the average relative percentage deviation in *TFT*:

$$\Delta_{average} = \frac{\sum_{i=1}^{R} \left(\frac{Heu_i - Best_{known}}{Best_{known}} \times 100 \right)}{R}$$

```
procedure VNS(x_0)
    x_best = x_0;                                   // x_best is the best solution obtained by VNS procedure
    x_current = x_0;
    do
        do
            x' = swap_local_search(x_current);
            x_current = insert_local_search(x');
        until (no possible improvement)
        if (TFT(x_current) < TFT(x_best)) then      // check the improvement of best solution
            x_best = x_current;
        endif
        x_current = x_best;
        find i and j randomly and permute the jobs in the positions i and j in x_current
    until (stopping criterion is found)
    return x_best;
end
```

Fig. 3. Pseudo code of VNS algorithm.

Table 1
Best known solutions

Instances	Best known	Algorithms	Instances	Best known	Algorithms	Instances	Best known	Algorithms
20 × 5	14033	PSOvns, H-CPSO, HGA, VNS, EDA–VNS	50 × 5	64817	EDA–VNS	100 × 5	254250	VNS
	15151	M-MMAS, PSOvns, H-CPSO, HGA, VNS, EDA–VNS		68066	VNS		243227	EDA–VNS
	13301	PSOvns, H-CPSO, HGA, VNS, EDA–VNS		63240	VNS		238580	VNS
	15447	PSOvns, H-CPSO, HGA, VNS, EDA–VNS		68287	EDA–VNS		228520	EDA–VNS
	13529	M-MMAS, PACO, PSOvns, H-CPSO, HGA, VNS, EDA–VNS		69478	VNS		241397	VNS
	13123	PACO, PSOvns, H-CPSO, HGA, VNS, EDA–VNS		66882	EDA–VNS		233161	VNS
	13548	PSOvns, H-CPSO, HGA, VNS, EDA–VNS		66274	EDA–VNS		241213	VNS
	13948	PSOvns, H-CPSO, HGA, VNS, EDA–VNS		64418	HGA		231865	VNS
	14295	PSOvns, H-CPSO, HGA,VNS, EDA–VNS		62981	VNS		249038	VNS
	12943	PSOvns, H-CPSO, HGA, VNS, EDA–VNS		68843	VNS		243647	EDA–VNS
20 × 10	20911	PSOvns, H-CPSO, HGA, VNS, EDA–VNS	50 × 10	87238	EDA–VNS	100 × 10	301001	EDA–VNS
	22440	M-MMAS, PSOvns, H-CPSO, HGA, VNS, EDA–VNS		83116	EDA–VNS		275601	VNS
	19833	M-MMAS, PSOvns, H-CPSO, HGA, VNS, EDA–VNS		80132	VNS		288943	VNS
	18710	PSOvns, H-CPSO, EDA–VNS		86725	VNS		303443	HGA
	18641	PSOvns, H-CPSO, HGA, VNS, EDA–VNS		86626	VNS		286646	HGA
	19245	M-MMAS, PACO, H-CPSO, EDA–VNS		86735	H-CPSO		271956	VNS
	18363	PSOvns, H-CPSO, HGA, VNS, EDA–VNS		89014	H-CPSO		281090	VNS
	20241	M-MMAS, PSOvns, H-CPSO, HGA, VNS, EDA–VNS		87025	EDA–VNS		293067	HGA
	20330	M-MMAS, PACO, PSOvns, H-CPSO, HGA, VNS, EDA–VNS		85688	EDA–VNS		303893	VNS
	21320	M-MMAS, PSOvns, H-CPSO, HGA, VNS, EDA–VNS		88149	H-CPSO		293492	EDA–VNS
20 × 20	33623	M-MMAS, H-CPSO, HGA, VNS, EDA–VNS	50 × 20	125831	VNS	100 × 20	368641	VNS
	31587	H-CPSO, HGA, VNS, EDA–VNS		119247	VNS, EDA–VNS		374838	EDA–VNS
	33920	M-MMAS, H-CPSO, HGA, VNS, EDA–VNS		116696	EDA–VNS		372423	EDA–VNS
	31661	H-CPSO, HGA, VNS, EDA–VNS		120834	EDA–VNS		374832	HGA
	34557	H-CPSO, HGA, VNS, EDA–VNS		118457	EDA–VNS		371268	HGA
	32564	H-CPSO, HGA, VNS, EDA–VNS		120820	EDA–VNS		375348	VNS
	32922	PACO, H-CPSO, HGA, VNS, EDA–VNS		123271	EDA–VNS		376353	HGA
	32412	H-CPSO, HGA, VNS, EDA–VNS		122820	EDA–VNS		387189	HGA
	33600	H-CPSO, HGA, VNS, EDA–VNS		121872	EDA–VNS		377729	VNS
	32262	H-CPSO, HGA, VNS, EDA–VNS		124486	EDA–VNS		381623	VNS

where Heu_i is the solution given by any of the R replications of the considered algorithms and $Best_{known}$ is the best known solution provided so far by an existing algorithm for the specified problem or by one of our proposed algorithms. Table 1 shows the best known solution for each instance and the corresponding algorithms that have provided it. We note that the bold numbers indicate that the best known solutions were provided by one of our proposed algorithms.

The heuristics used for benchmarking are the representative heuristics available in the literature of the PFSP with respect to the *TFT* criterion. These approaches consist of composite heuristics GA of Vempati et al. [15], BES (LR) of Liu and Reeves [9], ant colony algorithms (M-MMAS and PACO) of Rajendran and Ziegler [18], particle swarm algorithm (PSOvns) of Tasgetiren et al. [20], combinatorial particle swarm optimization algorithm (H-CPSO) of Jarboui et al. [19], hybrid genetic algorithm (HGA) of Zhang et al. [16] and the composite heuristic (ICH2) of Li et al. [14].

6.1. Comparison of the proposed EDA with GA

In this section, we perform a comparison of our EDA, without VNS algorithm, with a GA, thus, we have implemented the GA of Vempati et al. [15]. We note that, to the best of our knowledge, this is the only GA, without hybridization process, developed for the PFSP for minimizing the *TFT* criterion.

Here, the parameters of the EDA were fixed experimentally as follows: $P=60$, $\delta_1=\delta_2=4/n$, the number of the selected parents $Q=3$, the number of offspring generated $O=3$. The number of replications of each algorithm was set to $R=100$.

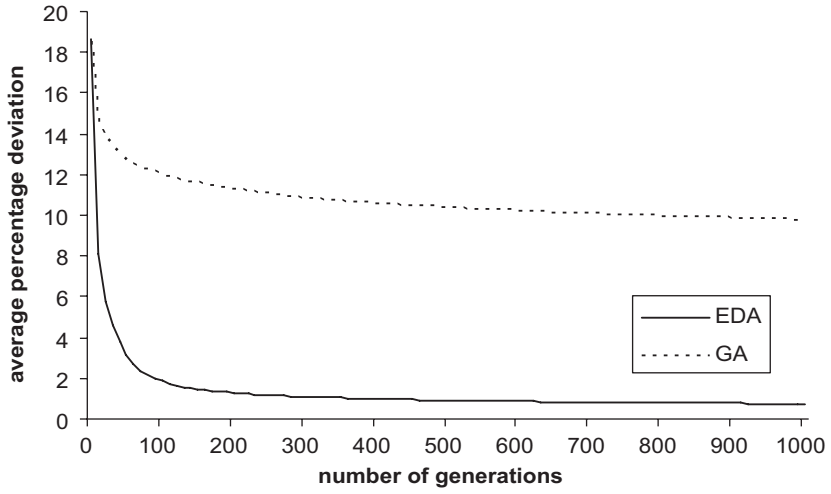

Fig. 4. Comparison results between EDA and GA for (20×5) instances.

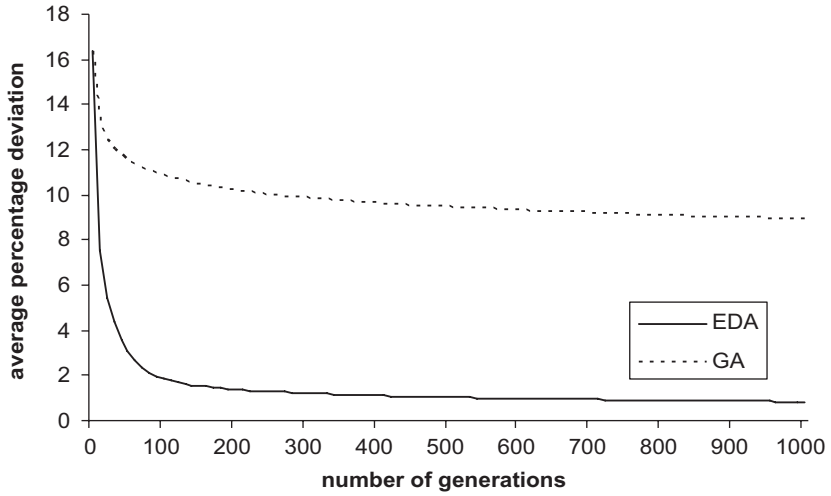

Fig. 5. Comparison results between EDA and GA for (20×10) instances.

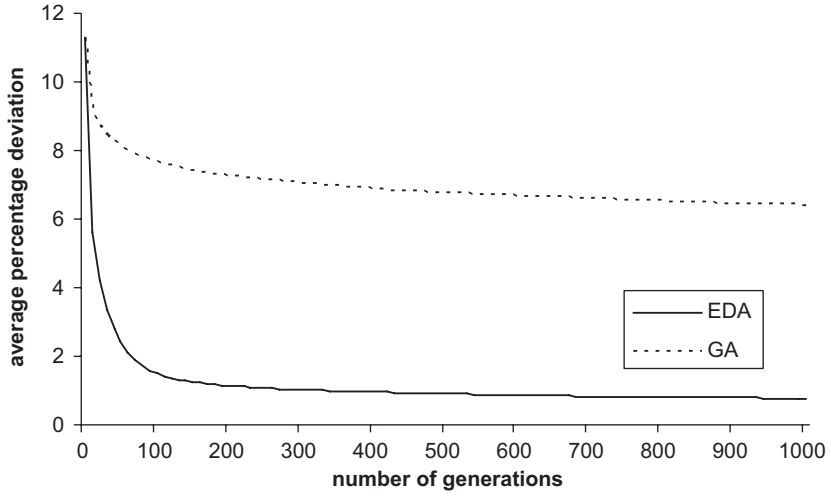

Fig. 6. Comparison results between EDA and GA for (20×20) instances.

Table 2
Comparison of EDA and GA

$n \times m$	EDA			CPU			GA			CPU		
	Δ_{min}	Δ_{avg}	Δ_{max}	Min	Average	Max	Δ_{min}	Δ_{avg}	Δ_{max}	Min	Average	Max
20×5	0.23	0.75	1.32	0.21	0.55	0.89	8.83	9.84	10.81	0.02	0.09	0.15
20×10	0.13	0.73	1.37	0.23	0.64	1.05	7.79	8.96	10.07	0.05	0.14	0.24
20×20	0.18	0.64	1.28	0.20	0.63	1.08	5.54	6.38	7.02	0.12	0.29	0.42
50×5	2.22	2.92	3.53	4.07	4.60	4.90	16.91	18.42	19.65	0.06	0.16	0.28
50×10	3.06	3.83	4.78	4.21	4.79	5.09	16.88	17.70	18.44	0.07	0.28	0.50
50×20	3.18	3.89	4.67	4.32	5.09	5.59	14.15	14.99	15.69	0.13	0.61	1.05
100×5	4.05	4.63	5.36	17.94	18.14	18.35	19.70	20.62	21.39	0.10	0.30	0.47
100×10	4.75	5.81	6.85	17.94	18.50	18.90	19.49	20.19	20.74	0.26	0.61	0.91
100×20	5.02	5.72	6.57	19.11	19.71	20.16	16.48	17.07	17.58	0.57	1.13	1.85
Average	2.54	3.22	3.97	7.58	8.07	8.45	13.97	14.91	15.70	0.15	0.40	0.65

Table 3
Results for the instances with $n = 20$, $m = 5, 10$ and 20

Instances	BES(LR)	M-MMAS	PACO	PSOvns	H-CPSO			VNS				EDA–VNS			
	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}
tai01	1.38	0.16	0.16	0.00	0.00	0.00	0.00	0.00	0.03	0.06	0.03	0.00	0.00	0.00	0.00
tai02	1.95	0.00	0.42	0.00	0.00	0.00	0.00	0.00	0.01	0.05	0.02	0.00	0.00	0.00	0.00
tai03	2.82	0.86	0.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai04	1.96	0.25	0.38	0.00	0.00	0.00	0.00	0.00	0.02	0.08	0.03	0.00	0.00	0.00	0.00
tai05	0.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai06	1.08	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai07	1.67	0.08	0.93	0.00	0.00	0.04	0.07	0.00	0.05	0.08	0.04	0.00	0.00	0.00	0.00
tai08	0.14	0.14	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai09	1.13	0.15	0.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai10	0.72	0.19	0.60	0.00	0.00	0.00	0.00	0.00	0.10	0.25	0.14	0.00	0.00	0.00	0.00
Average	1.36	0.20	0.45	0.00	0.00	0.00	0.01	0.00	0.02	0.05		0.00	0.00	0.00	
tai11	1.42	0.33	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai12	2.17	0.00	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai13	1.21	0.00	0.68	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai14	0.79	0.07	0.32	0.00	0.00	0.00	0.00	0.07	0.19	0.22	0.06	0.00	0.00	0.00	0.00
tai15	1.60	0.02	0.58	0.00	0.00	0.00	0.00	0.00	0.08	0.20	0.08	0.00	0.00	0.00	0.00
tai16	1.89	0.00	0.00	0.02	0.00	0.00	0.02	0.02	0.39	0.55	0.22	0.00	0.00	0.00	0.00
tai17	1.96	0.07	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai18	1.30	0.00	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai19	1.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai20	0.87	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.04	0.08	0.03	0.00	0.00	0.00	0.00
Average	1.43	0.05	0.32	0.00	0.00	0.00	0.00	0.01	0.07	0.10		0.00	0.00	0.00	
tai21	1.48	0.00	0.00	4.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai22	1.05	0.05	0.03	3.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai23	1.86	0.00	0.62	1.99	0.00	0.00	0.00	0.00	0.06	0.15	0.08	0.00	0.00	0.00	0.00
tai24	1.57	0.12	0.29	3.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai25	1.25	0.10	0.25	2.60	0.00	0.00	0.00	0.00	0.05	0.08	0.05	0.00	0.00	0.00	0.00
tai26	0.52	0.22	0.09	2.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai27	1.60	0.35	0.00	2.46	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai28	0.61	0.10	0.37	1.84	0.00	0.00	0.00	0.00	0.04	0.10	0.05	0.00	0.00	0.00	0.00
tai29	1.44	0.07	0.07	2.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai30	0.85	0.17	0.17	3.16	0.00	0.00	0.00	0.00	0.08	0.17	0.09	0.00	0.00	0.00	0.00
Average	1.22	0.12	0.19	2.83	0.00	0.00	0.00	0.00	0.02	0.05		0.00	0.00	0.00	

Figs. 4–6 present the results obtained by both algorithms in the generation space for the benchmarks with 20 jobs ($(20 \times 5), (20 \times 10)$ and (20×20)). They report the average percentage deviation according to the number of generations for EDA and GA. In fact, the EDA outperforms the GA of Vempati et al. [15] in term of solution quality. Furthermore, EDA can provide high quality solutions rapidly over the first 200 generations.

Table 2 shows the results obtained by EDA and GA after 1000 generations tested on the 90 benchmark problems from Taillard. Although EDA is better than GA of Vempati et al. [15] in term of solution quality, the GA appears more efficient in terms of CPU time because it requires a linear time to create new individual, whereas this task requires $O(n^2)$ time for the EDA.

6.2. Relative performances of VNS and EDA–VNS

In this section, we discuss the relative performances of our proposed algorithms (VNS, EDA–VNS) compared with existing approaches. All the parameters of the algorithms are set experimentally: $P = 10$, δ_1 and δ_2 are equal to $4/n$, $Q = 3$ and $O = 3$. We fixed parameter α according to the relative deviation RD between the current solution and the best solution found by the algorithm and the

Table 4
Results for the instances with $n = 20$, $m = 5$, 10 and 20

Instances	BES(LR)	M-MMAS	PACO	PSOvns	H-CPSO			VNS				EDA–VNS			
	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}
tai31	1.31	1.47	1.12	0.37	0.03	0.15	0.28	0.04	0.16	0.35	0.12	0.00	0.08	0.20	0.07
tai32	0.88	1.12	0.62	0.34	0.23	0.38	0.52	0.00	0.13	0.22	0.09	0.12	0.17	0.20	0.03
tai33	1.80	1.46	1.44	0.53	0.31	0.53	0.69	0.00	0.32	0.55	0.24	0.03	0.17	0.49	0.18
tai34	2.21	1.21	1.57	0.42	0.44	0.62	0.78	0.25	0.35	0.57	0.13	0.00	0.12	0.27	0.11
tai35	1.96	1.23	0.97	0.32	0.15	0.30	0.40	0.00	0.16	0.37	0.13	0.02	0.07	0.13	0.04
tai36	1.79	1.02	1.17	0.38	0.27	0.41	0.60	0.17	0.27	0.35	0.08	0.00	0.13	0.27	0.12
tai37	1.38	1.12	0.49	0.10	0.15	0.28	0.52	0.09	0.25	0.74	0.28	0.00	0.07	0.15	0.07
tai38	1.81	0.69	1.09	0.34	0.18	0.42	0.61	0.01	0.26	0.38	0.15	0.02	0.14	0.2	0.07
tai39	1.57	1.20	0.80	0.39	0.28	0.42	0.56	0.00	0.20	0.34	0.12	0.12	0.18	0.23	0.05
tai40	2.08	2.05	1.44	0.51	0.40	0.62	0.78	0.00	0.30	0.60	0.22	0.17	0.30	0.47	0.13
Average	1.68	1.26	1.07	0.37	0.24	0.41	0.57	0.06	0.24	0.45		0.05	0.14	0.26	
tai41	1.76	2.71	1.95	0.91	0.50	0.99	1.69	0.07	0.58	1.20	0.41	0.00	0.22	0.59	0.22
tai42	2.99	0.60	1.72	0.61	0.10	0.83	1.43	0.09	0.54	0.79	0.30	0.00	0.25	0.36	0.14
tai43	2.90	1.90	1.51	0.60	0.22	0.51	1.05	0.00	0.34	0.52	0.21	0.15	0.23	0.29	0.06
tai44	3.03	1.38	1.49	0.38	0.36	0.48	0.74	0.00	0.17	0.29	0.11	0.14	0.19	0.24	0.04
tai45	2.14	2.54	1.36	0.74	0.22	0.65	1.31	0.00	0.38	0.85	0.31	0.06	0.16	0.25	0.10
tai46	3.31	1.91	1.77	0.60	0.00	0.51	0.84	0.06	0.21	0.33	0.10	0.05	0.18	0.26	0.09
tai47	2.71	1.88	1.09	0.28	0.00	0.44	0.84	0.20	0.51	0.85	0.26	0.26	0.36	0.48	0.08
tai48	2.90	1.80	1.44	0.19	0.36	0.56	0.88	0.11	0.31	0.54	0.16	0.00	0.20	0.39	0.15
tai49	2.97	1.50	1.53	0.48	0.32	0.54	0.98	0.14	0.33	0.60	0.17	0.00	0.20	0.41	0.16
tai50	2.99	1.50	1.24	0.55	0.00	0.76	0.85	0.19	0.43	0.71	0.23	0.21	0.28	0.42	0.08
Average	2.77	1.77	1.51	0.53	0.21	0.63	1.06	0.09	0.38	0.67		0.09	0.23	0.37	
tai51	2.59	1.21	0.90	2.22	0.23	0.70	1.02	0.00	0.52	1.10	0.47	0.24	0.32	0.51	0.11
tai52	2.39	1.64	1.55	2.45	0.58	0.76	1.06	0.00	0.48	1.05	0.52	0.00	0.20	0.28	0.12
tai53	4.01	1.16	0.71	1.73	0.44	0.56	0.74	0.22	0.40	0.74	0.21	0.00	0.14	0.33	0.13
tai54	2.69	1.84	1.63	1.82	0.58	0.74	1.17	0.12	0.42	0.57	0.18	0.00	0.15	0.22	0.09
tai55	3.12	1.24	0.64	2.32	0.28	0.64	1.20	0.32	0.56	0.82	0.21	0.00	0.15	0.28	0.12
tai56	2.68	1.28	1.19	1.98	0.08	0.40	0.65	0.03	0.39	0.83	0.29	0.00	0.16	0.26	0.10
tai57	2.51	2.14	1.69	1.88	0.39	0.61	0.85	0.21	0.30	0.42	0.08	0.00	0.14	0.38	0.15
tai58	2.85	1.40	1.27	2.36	0.07	0.53	1.03	0.12	0.48	0.84	0.32	0.00	0.07	0.15	0.07
tai59	2.83	1.79	1.46	2.51	0.34	0.74	1.22	0.36	0.59	0.77	0.19	0.00	0.26	0.43	0.17
tai60	2.68	1.68	1.03	1.47	0.03	0.27	0.49	0.04	0.34	0.58	0.24	0.00	0.10	0.18	0.07
Average	2.84	1.54	1.21	2.07	0.30	0.60	0.94	0.14	0.45	0.77		0.02	0.17	0.30	

calculated probability p^c. Numerically, $p^c = 0.5$ leads to accepting a sequence with a *TFT* superior by 1% relatively to the best value of *TFT* found. So, $\alpha = RD/\log(p^c) = 0.01/\log(0.5)$, thereafter we determined p^c according to this formula: $p^c = \max\{\exp(RD/\alpha), \varepsilon\}$ with $\varepsilon = 0.01$. The maximum number of iterations of VNS algorithm ($iter_{\max}$) was set at 50. In our case, we set a maximal computational time equal to $n \times m \times 0.4$ seconds as a stopping criterion. We set the number of replications of our algorithms to $R = 5$, similarly to the compared approaches.

Tables 1–6 show the results provided by our proposed approaches (VNS and EDA–VNS) compared with the results given by BES (LR) of Liu and Reeves [9], ant colony algorithms (M-MMAS and PACO) of Rajendran and Ziegler [18], particle swarm optimization algorithm (PSOvns) of Tasgetiren et al. [20] and combinatorial particle swarm optimization algorithm (H-CPSO) of Jarboui et al. [19].

As seen in Table 1, our VNS and EDA–VNS have in total improved 49 solutions among 90. Therefore, 26 out of 49 best known solutions were generated by VNS algorithm, whereas 24 were generated by EDA–VNS. Moreover, both algorithms were able to reach all best known solutions. In addition, most significantly improved solutions of EDA–VNS occur in large instances ($n = 50$ and 100) with 50 improved instances out of 60.

Tables 3–5 present the minimum, average and maximum deviations of different approaches, over 5 runs, for the instances with $n = 20$, $n = 50$ and $n = 100$, respectively. In average, it was shown through Δ_{max} performance measure that the worst solutions provided by EDA–VNS are better than or equal to the best solutions found by BES(LR), MMAS and PACO over 5 runs. Also, Δ_{max} of EDA–VNS is less than or equal to the best results of PSOvns and H-CPSO, for 78 instances and 68 instances, respectively. Table 3 shows that EDA–VNS can reach the best solutions over the five runs for all instances with $n = 20$. This result proves the robustness of EDA–VNS for the instances with small sizes.

By comparing EDA–VNS and VNS, it was shown that, in average, EDA–VNS performs better than VNS algorithm for the instances with a size lower than 100 jobs, but for the remaining sizes (Table 5), VNS outperforms EDA–VNS for the instances with $m = 5$ and 10 and these two algorithms are very close for $m = 20$. In term of $\Delta_{\min}$, in average, the EDA–VNS and VNS results are globally very similar. The standard deviation values of the relative percentage of difference in the solutions (Δ_{std}) are provided to give an idea about the robustness of the algorithms. Δ_{std} values of EDA–VNS are smaller than the ones of VNS for all experimental runs (Tables 3–5), thus the hybridization process provides more stability to the algorithm. Concerning the CPU times, VNS and EDA–VNS appear similar in average (Table 6).

Now, we present the comparative results between our EDA–VNS and HGA of Zhang et al. [16] and ICH2 of Li et al. [14]. This comparative study was based on the values of $\Delta_{average}$ computed according to the upper bounds given in Zhang et al. [16]. Table 7 summarizes the results found using EDA–VNS, HGA and ICH2, in term of the average value, for each instance size. It appears that EDA–VNS outperforms HGA and ICH2 for all sizes of instances.

7. Conclusion

An EDA for the PFSP was presented in this work with the objective to minimize the *TFT*. The initial population is generated randomly. The probabilistic model built focuses on both the importance of the

Table 5
Results for the instances with $n = 50$, $m = 5$, 10 and 20

Instances	BES(LR)	M-MMAS	PACO	PSOvns	H-CPSO			VNS				EDA–VNS			
	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}
tai61	1.00	1.09	1.43	0.20	0.50	0.72	0.90	0.00	0.18	0.47	0.18	0.24	0.36	0.55	0.12
tai62	0.98	1.39	1.27	0.86	0.53	0.83	0.98	0.06	0.20	0.39	0.14	0.00	0.24	0.45	0.19
tai63	1.02	0.82	1.13	0.50	0.53	0.76	0.92	0.00	0.15	0.32	0.12	0.10	0.19	0.29	0.08
tai64	1.24	0.86	0.81	0.15	0.42	0.60	0.69	0.01	0.04	0.09	0.03	0.00	0.11	0.29	0.12
tai65	1.08	0.67	0.85	0.35	0.34	0.53	0.65	0.00	0.13	0.25	0.10	0.05	0.13	0.23	0.06
tai66	1.13	1.31	1.39	0.40	0.53	0.72	0.80	0.00	0.27	0.46	0.18	0.24	0.47	0.67	0.15
tai67	1.05	1.13	1.09	0.38	0.65	0.78	0.93	0.00	0.24	0.45	0.17	0.02	0.20	0.30	0.11
tai68	1.43	1.27	1.17	0.38	0.44	0.56	0.77	0.00	0.18	0.39	0.16	0.10	0.22	0.36	0.11
tai69	0.90	1.34	1.72	0.37	0.50	0.59	0.66	0.00	0.13	0.24	0.11	0.05	0.15	0.28	0.10
tai70	1.58	1.07	1.27	0.26	0.52	0.70	0.86	0.10	0.17	0.27	0.06	0.00	0.32	0.50	0.22
Average	1.14	1.10	1.22	0.39	0.50	0.68	0.82	0.02	0.17	0.33		0.08	0.24	0.39	
tai71	1.79	1.33	1.45	0.71	0.65	0.87	1.02	0.14	0.31	0.64	0.21	0.00	0.14	0.22	0.09
tai72	1.93	1.27	1.20	0.55	0.66	1.15	1.40	0.00	0.48	1.23	0.57	0.17	0.55	0.84	0.25
tai73	2.76	2.85	1.83	1.22	0.94	1.18	1.59	0.00	0.45	0.73	0.30	0.35	0.63	0.74	0.16
tai74	2.03	1.17	1.09	0.41	0.73	1.05	1.37	0.05	0.14	0.32	0.11	0.04	0.25	0.58	0.23
tai75	1.77	1.34	1.06	0.56	0.39	0.72	0.89	0.09	0.28	0.43	0.14	0.19	0.27	0.33	0.06
tai76	1.76	1.65	1.46	0.31	0.81	0.99	1.21	0.00	0.20	0.34	0.13	0.04	0.23	0.34	0.12
tai77	2.53	1.94	1.34	0.48	0.54	0.92	1.21	0.00	0.43	0.68	0.30	0.36	0.44	0.52	0.06
tai78	0.91	1.36	1.34	0.04	0.67	0.89	1.06	0.00	0.15	0.37	0.15	0.13	0.26	0.38	0.10
tai79	2.73	1.90	1.04	0.56	0.63	0.93	1.31	0.00	0.24	0.39	0.15	0.19	0.40	0.55	0.15
tai80	1.84	1.15	1.26	0.57	1.00	1.24	1.70	0.03	0.26	0.44	0.19	0.00	0.20	0.27	0.11
Average	2.02	1.61	1.32	0.55	0.72	1.01	1.29	0.04	0.31	0.57		0.16	0.35	0.49	
tai81	4.13	1.39	1.08	1.55	1.04	1.19	1.33	0.00	0.38	0.89	0.33	0.20	0.47	0.72	0.21
tai82	2.44	2.34	1.68	1.32	0.44	1.19	1.83	0.34	0.56	0.74	0.15	0.00	0.46	0.69	0.28
tai83	3.05	2.06	1.80	1.54	0.89	1.12	1.51	0.18	0.31	0.36	0.07	0.00	0.27	0.52	0.21
tai84	2.67	1.43	1.58	1.62	1.18	1.41	1.71	0.19	0.68	1.24	0.39	0.15	0.42	0.64	0.20
tai85	3.38	1.62	2.10	1.32	0.85	1.31	1.72	0.05	0.29	0.66	0.23	0.33	0.47	0.58	0.10
tai86	3.36	1.64	1.36	1.04	0.81	0.93	1.06	0.00	0.23	0.50	0.20	0.05	0.17	0.41	0.15
tai87	2.25	1.49	1.50	1.21	0.81	1.01	1.25	0.00	0.48	1.32	0.53	0.21	0.45	0.63	0.16
tai88	2.52	1.58	1.44	1.25	0.80	1.01	1.26	0.14	0.25	0.37	0.10	0.00	0.31	0.52	0.19
tai89	2.67	2.05	0.70	1.19	0.80	0.93	1.17	0.00	0.37	0.64	0.26	0.09	0.39	0.75	0.26
tai90	3.47	1.66	1.69	1.15	0.82	0.97	1.25	0.00	0.35	0.61	0.27	0.26	0.39	0.58	0.13
Average	3.00	1.73	1.50	1.33	0.85	1.12	1.42	0.10	0.40	0.74		0.14	0.39	0.61	

Table 6
Computational times of the VNS and EDA–VNS algorithms

Instances	VNS	EDA–VNS
20×5	0.13	0.30
20×10	0.30	1.29
20×20	0.74	1.51
50×5	31.98	57.22
50×10	56.18	105.45
50×20	142.08	240.96
100×5	174.26	124.55
100×10	324.37	266.02
100×20	644.98	570.27
Average	152.78	151.95

Table 7
Comparison of EDA–VNS with ICH2 and HGA

Instances	ICH2	HGA	EDA–VNS
20×5	1.034	0.025	0.000
20×10	1.308	0.055	0.000
20×20	1.447	0.033	0.000
50×5	1.275	0.144	−0.120
50×10	1.927	0.222	−0.081
50×20	2.395	0.288	−0.084
100×5	0.765	0.125	0.025
100×10	1.512	0.230	0.154
100×20	2.353	0.336	0.270
Average	1.557	0.162	0.018

order of the jobs in the sequences and the similar blocks of jobs presented in the selected parents. After creating new individuals based on the estimated model, we introduced an improvement procedure by using a VNS algorithm. The experiments show that our algorithms outperform other methods employed for the problem. For the small benchmarks, EDA–VNS is better than VNS but, for the large ones, VNS is superior both in terms of solution quality and computational time.

References

[1] Gupta JND, Stafford Jr. EF. Flowshop scheduling research after five decades. European Journal of Operational Research 2006;169:699–711.
[2] Garey MR, Johnson DS, Sethi R. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1976;2:117–29.
[3] Chung CS, Flynn J, Kirca O. A branch and bound algorithm to minimize the total flow time for m-machine permutation flowshop problems. International Journal of Production Economics 2002;79:185–96.
[4] Van de Velde SL. Minimizing the sum of the job completion times in the two-machine flow shop by Lagrangian relaxation. Annals of Operations Research 1990;26:257–68.
[5] Bansal SP. Minimizing the sum of completion times of n jobs over m machines in a flowshop: a branch and bound approach. AIIE Transactions 1977;9:306–11.
[6] Ignall E, Schrage L. Application of the branch and bound technique to some flow-shop scheduling problems. Operations Research 1965;13:400–12.
[7] Woo HS, Yim DS. A heuristic algorithm for mean flowtime objective in flowshop scheduling. Computers and Operations Research 1998;25:175–82.
[8] Framinan JM, Leisten R. An efficient constructive heuristic for flowtime minimisation in permutation flow shops. Omega 2003;31:311–7.
[9] Liu J, Reeves CR. Constructive and composite heuristic solutions to the $P|\sum C_i$ scheduling problem. European Journal of Operational Research 2001;132:439–52.
[10] Ho CJ, Gupta JND. Flowshop scheduling with dominant machines. Computers and Operations Research 1995;22:237–46.

[11] Framinan JM, Leisten R, Ruiz-Uzano R. Comparison of heuristics for flowtime minimisation in permutation flowshops. Computers and Operations Research 2005;32:1237–54.
[12] Rajendran C. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. International Journal of Production Economics 1993;29:65–73.
[13] Allahverdi A, Aldowaisan T. New heuristics to minimize total completion time in m-machine flowshops. International Journal of Production Economics 2002;77:71–83.
[14] Li X, Wang Q, Wu C. Efficient composite heuristics for total flowtime minimization in permutation flowshops. Omega 2009;37:155–64.
[15] Vempati VS, Chen CL, Bulllngton SF. An Effective Heuristic for Flow Shop Problems with Total Flow Time as Criterion. Computers and Industrial Engineering 1993;25:219–22.
[16] Zhang Y, Li X, Wang Q. Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. European Journal of Operational Research (2008) doi: 10.1016/j.ejor.2008.04.033.
[17] Gajpal Y, Rajendran C. An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops. International Journal of Production Economics 2006;101:259–72.
[18] Rajendran C, Ziegler H. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. European Journal of Operational Research 2004;155:426–38.
[19] Jarboui B, Ibrahim S, Siarry P, Rebai A. A combinatorial particle swarm optimisation for solving permutation flowshop problems. Computers and Industrial Engineering 2008;54:526–38.
[20] Tasgetiren MF, Liang Y-C, Sevkli M, Gencyilmaz G. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. European Journal of Operational Research 2007;177:1930–47.
[21] Mühlenbein H. Paaß G. From recombination of genes to the estimation of distribution. Binary parameters. In: Lecture notes in computer science 1411: parallel problem solving from nature, PPSN, 1996; IV: p. 178–87.
[22] Larraanaga, P., Lozano, J.A., (Eds.). Estimation of distribution algorithms: a new tool for evolutionary computation 2002. Boston/Dordrecht/London: Kluwer Academic Publishers; 2002.
[23] Lozano JA, Larranaga P, Inza I, Bengoetxea E. Towards a New Evolutionary Computation Advances on Estimation of Distribution Algorithms. Berlin: Springer; 2006.
[24] Mladenović N, Hansen P. Variable neighborhood search. Computers and Operations Research 1997;24:1097–100.
[25] Hansen P, Mladenović N. Variable neighborhood search: principles and applications. European Journal of Operational Research 2001;130:449–67.
[26] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 1979;5:287–326.
[27] Reeves CR. A genetic algorithm for flowshop sequencing. Computers and Operations Research 1995;22:5–13.
[28] Taillard E. Benchmarks for basic scheduling problems. European Journal of Operational Research 1993;64:278–85.