An Estimation of Distribution Algorithm for Minimizing the Makespan in Blocking Flowshop Scheduling Problems

Bassem Jarboui[1], Mansour Eddaly[1], Patrick Siarry[2], and Abdelwaheb Rebaï[1]

[1] FSEGS, route de l'aéroport km 4.5, B.P. No. 1088, Sfax 3018, Tunisie
bassem_jarboui@yahoo.fr, eddaly.mansour@gmail, abdelwaheb.rebai@fsegs.rnu.tn
[2] LiSSi, Université de Paris 12, 61 avenue du Général de Gaulle, 94010 Créteil, France
siarry@univ-paris12.fr

Summary. This chapter addresses to the blocking flowshop scheduling problem with the aim of minimizing the makespan. An Estimation of Distribution Algorithm, followed by a local search procedure, after the step of creating a new individual, was developed in order to solve this problem. Our comparisons were performed against representative approaches proposed in the literature related to the blocking flowshop scheduling problem. The obtained results have shown that the proposed algorithm is able to improve 109 out of 120 best known solutions of Taillard's instances. Moreover, our algorithm outperforms all competing approaches in terms of solution quality and computational time.

1 Introduction

In the nature, the evolution of species in a population, through the sexual reproduction, was formulated by Charles Darwin (T. Back, 1996). It can be modelled by means of three mechanisms: recombination (or crossover), mutation and selection. The process of recombination occurs during meiosis resulting from crossover between parental chromosomes. Through this process, the offspring inherit different combinations of genes from their parents. The mutation arises from errors of copying in genetic materials during cell division. It creates changes into offspring's chromosomes. Under selection, individuals with best traits tend to have more luck to survive and reproduce for further generations. Evolutionary algorithms (EAs) are a class of algorithms that use computers to simulate the natural evolution of species to solve hard optimization problems through evolving a population of candidate solutions. EAs have proved their performance against classical techniques of optimization (Fogel, 1995). Several algorithms are included in this class such as the Genetic Algorithm (GA), which is the most popular. Neighbouring nature-inspired approaches are Ant Colony Optimization, Particle Swarm Optimization, etc.

Recently, a new EA was introduced by Mühlenbein and Paaß in (Mühlenbein and Paaß, 1996), called Estimation of Distribution Algorithm (EDA). It constitutes a new tool of evolutionary algorithms (Larranaga P. and Lozano J.A., 2002), based on the probabilistic model learned from a population of individuals. Starting with a population of individuals (candidate solutions), generally randomly generated, 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 are generated from the estimated distribution. The process is repeated until the termination criterion is met. In the literature, diverse versions of EDAs were developed, depending on the chosen probabilistic model. The EDAs can be classified into three classes: EDAs with no dependencies between the variables, EDAs with two-order dependencies and EDAs with multiple dependencies between the variables.

EDAs have been employed for solving combinatorial optimization problems. So, several successful applications were proposed such as: quadratic assignment problem (Zhang et al., 2006), 0-1 knapsack problem (Hui Li et al., 2004), n-queen problem (Paul TK and Iba H, 2002), travelling salesman problem (Robles et al., 2006) and hybrid flowshop scheduling problem (Salhi et al., 2007). In recent works, the EDAs were devoted to solve multi-objective optimization problems (Zhang et al. 2008, Hui Li et al., 2004).

In this work, we propose to adopt this new technique for solving the blocking flowshop scheduling problem. In this variant of flowshop scheduling, there is a set of n jobs that must be processed on a set of m machines in the same order. While the storage is not allowed, when a job is completed on a machine, the latter is blocked until a free next machine becomes available. Blocking constraints takes place because of the automation of new production systems and the use of the robotic manufacturing. Typical areas are chemical and pharmaceutical industries, where a partially completed job cannot quit the machine on which it is processed, while downstream machines are busy (Grabowski and Pempera, 2007). Grabowski and Pempera (2000) have presented a real case of scheduling client orders in a building industry that produces concrete blocks. Also, Hall and Sriskandarajah (1996) have presented a review of applications of blocking scheduling models. They have indicated that blocking environment occurs from characteristics of the process technology itself or from the lack of the storage capacity between the machines. They have proved that this problem is strongly NP-complete for m=3, where the makespan $\left(C_{\max}\right)$ is a measure of performance.

In the literature, various approaches were developed to solve the permutation flowshop scheduling problem under blocking constraints, including branch and bound algorithm (B&B) (Levner, 1969, Suhami and Mah, 1981, Ronconi, 2005, Company and Mateo, 2007), constructive heuristics (McCormick et al., 1989, Leisten, 1990, Abadi et al., 2000, Ronconi and Armentano, 2001, Ronconi, 2004), genetic algorithm (GA) (Caraffa et al., 2001) and tabu search (TS) (Grabowski and Pempera, 2007).

The remaining of this chapter is organized as follows: section 2 presents the Estimation of Distribution Algorithm and its variants; section 3 presents the existing works with EDA in combinatorial optimization. The blocking flowshop is described in section 4. Our proposed algorithm is presented in section 5. Section 6 presents the computational results and conclusion is given in section 7.

2 Estimation of Distribution Algorithm (EDA)

EDA is an evolutionary algorithm proposed by Mühlenbein and Paaß in 1996. Instead of recombination and mutation, EDA generates new individuals with respect to a probabilistic model, learned from the population of parents.

2.1 Basic EDA

The general framework of the basic EDA can be presented as follows (Mühlenbein and Paaß, 1996). Starting with a randomly generated initial population, one selects a subpopulation of M parent individuals through a selection method based on the fitness function. Next, one estimates the probability of distribution of the selected parents with a probabilistic model. Then, one generates new offspring, according to the estimated probability distribution. Finally, some individuals in the current population are replaced with new generated offspring. These steps are repeated until one stopping criterion is met. The pseudo-code of the canonical EDA is given in Figure 1.

```
Basic EDA

Generate an initial population of P individuals;
do
    • Select a set of $Q$ parents with a selection method;
    • Build a probabilistic model for the set of selected parents;
    • Create new $P_1$ offspring according to the estimated probability distribution;
    • Replace some individuals in the current population with new individuals;
while a stopping criterion is not met
```

Fig. 1. Canonical version of EDA

Three classes of EDA were developed, according to the chosen probabilistic model. The first class consists of models which don't take into account the dependencies between variables of candidate solutions, i.e. all variables are independent. The second class assumes at most two-order dependencies between these variables and the last class assumes multiple dependencies between the variables.

2.2 EDAs with No Dependencies

Let X_i, $i = 1, 2, ..., n$, be a random variable and x_i its possible realization and let $p(X_i = x_i) = p(x_i)$ the mass probability of X_i over the point x_i. By analogy, we denote by $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ a set of n-dimensional random variables, $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ its possible realizations and $p(\mathbf{X} = \mathbf{x}) = p(\mathbf{x})$ the joint mass probability of $\mathbf{X}$ over the point $\mathbf{x}$.

In this class of EDAs, it is assumed that the n-dimensional joint probability distribution is calculated through the product of the marginal probabilities of n variables, as follows:

$$p(x) = \prod_{i=1}^{n} p(x_i).$$

In other hand, the hypothesis of interaction between the variables is rejected.

Among the EDAs included in this class we can cite: Bit-Based Simulated Crossover (BBSC) of Syswerda (1993), Population-Based Incremental Learning (PBIL) of Baluja (1994), Compact Genetic Algorithm (CGA) of Harik et al. (1998) and Univariate Marginal Distribution Algorithm (UMDA) of Mühlenbein et al. (1998).

Although these approaches have provided better results for some problems, their assumption seems to be inexact for difficult optimization problems, where we cannot exclude the interdependencies between the variables completely (Paul TK and Iba H, 2002).

2.3 EDAs with Two-Order Dependencies

In this class, only paired interactions between the variables are taken into account. So, EDAs belonging to this group constitute an extension of the previous one. Therefore, the parametric learning of model, proposed in EDAs with no interaction, becomes structural.

In the literature, several approaches were developed in this class, such as: Mutual Information Maximization for Input Clustering (MIMIC) in De Bonet al. (1997), Combining Optimizers with Mutual Information Trees (COMIT) in Baluja and Davies (1997) and Bivariate Marginal Distribution Algorithm (BMDA) in Pelikan and Mühlenbein (1999).

2.4 EDAs with Multiple Dependencies

This last class of EDAs is the most general case, and the leaning process of models proposed here is more complex, because the estimation of joint probability is performed by taking into account an order of dependencies greater than two.

The following approaches of EDAs are included in this class: Factorized Distribution Algorithm (FDA) (Mühlenbein et al., 1999), Estimation of Bayesian Networks Algorithm (EBNA) (Etxeberria and Larranaga, 1999), Bayesian Optimization Algorithm (BOA) (Pelikan et al., 1999), Learning Factorized Distribution Algorithm (LFDA) (Mühlenbein and Mahning, 1999) and the Extended Compact Genetic Algorithm (ECGA) (Harik, 1999).

3 Some EDAs for Combinatorial Optimization Problems

Although, EDA was recently invented, the number of its applications in the field of combinatorial optimization increases rapidly. In this section, we will present some applications of EDA to combinatorial optimization problems and we will mainly focus on the constructed probabilistic model for each application.

The Jobshop Scheduling Problem (JSP) was addressed by J. Lozano et al. (in Larrañaga and Lozano, 2002). The authors have selected some variants of EDA and used both continuous and discrete versions. The selected algorithms are UMDA, BBSC, PBIL, MIMIC and EBNA.

The obtained results are comparable to those obtained using GA. In particular the continuous EDAs perform better than the discrete EDAs.

Paul TK and Iba H have proposed, in (Paul TK and Iba H, 2002), an UMDA to solve *n*-queen problem. The objective of this problem is to find a way of putting n_q queens ($n_q \geq 4$) on a $n_q \times n_q$ chessboard, such that none of them can capture any other, i.e. two queens cannot share the same row, column or diagonal. A problem's solution *x* was represented as follows: $\mathbf{x} = \{x_1, x_2, ..., x_{nq}\}$, where x_i, $1 \leq i \leq n_q$, denotes the column position in row *i* where the queen *i* can be put. The initial population was randomly generated while excluding cases where two queens are in the same column or row. The fitness of each individual is calculated as the number of queens that do not share the same diagonal. Next, the first 50% of individuals (best individuals) were selected according to their fitness. Then, the joint probability was selected using the marginal frequencies of each x_i and new individuals were generated according to it. Finally, the elitism was used for the replacement step and the algorithm was stopped when the fitness of the best individual was equal to n_q. The computational results show that this algorithm is able to reach a good solution in a reasonable amount of time.

Hui et *al.* (2004) have proposed a hybrid EDA for solving the multiobjective 0-1 knapsack problem. For modelling the joint probability distribution, an UMDA is used. At each generation *t*, an individual is selected, based on the following probability, depending on the set of selected individuals at generation *t*-1:

$$p(x,t) = p\left(x/\text{selected individuals}(t-1)\right) = \prod_{i=1}^{n_k} p(x_i,t)$$

where $x \in \{0,1\}^{n_k}$.

The results showed that the EDA performed better than the Genetic Algorithm, both in convergence and in diversity.

Salhi et *al.* (2007) have proposed an EDA for hybrid flowshop scheduling problem with respect to the makespan criterion. The joint probability $p_{ij}(t)$ denotes the probability that the job *i* is located on the position *j* at the generation *t* $(1 \leq i \leq n \text{ and } 1 \leq j \leq n)$.

This probability was initially set to $1/n^2$ and updated as follows:

$$p_{ij}(t) = (1-\beta)\frac{1}{N}\sum_{k=1}^{N} I_{ij}(\pi_k) + \beta p_{ij}(t-1)$$

where π_k is the k^{th} solution of the population at the generation *t* $(1 \leq k \leq N)$, $I_{ij} = \begin{cases} 1 & if\, \pi(i) = j \\ 0 & otherwise \end{cases}$ and $(0 \leq \beta \leq 1)$.

The obtained results were compared with those provided by two heuristic algorithms, a Random Key Genetic Algorithm and a Genetic Algorithm. The results show that EDA outperforms these two algorithms for the considered instances.

4 Problem Description

In a blocking flowshop problem, there is a set of *n* jobs to be processed on a set of *m* machines in the same order, while having no intermediate buffers, i.e. a job $j \in \{1, 2, ..., n\}$ cannot pass from machine $k \in \{1, 2, ..., m\}$ to machine k+1 while the latter is busy. Since the makespan is the criterion to be minimized in our case, this problem can be denoted by $F_m / blocking / C_{\max}$ (Graham et al., 1979).

Let $p_{[j]k}$ denote the processing time of the job in the *j*th position in the sequence on the machine *k* and $D_{[j]k}$ denote the departure time (starting time) of the job in the *j*th position in the sequence on the machine *k*.

The makespan $(C_{\max})$ can be found through the recursive expression of the departure time, as follows:

$$
\begin{aligned}
&D_{[1]0} = 0; \\
&D_{[1]k} = \sum_{i=1}^{k} p_{[1]i} \qquad k = 1, 2, ..., m-1; \\
&D_{[j]0} = D_{[j-1]1} \qquad j = 2, 3, ..., n; \\
&D_{[j]k} = \max\left\{ D_{[j]k-1} + p_{[j]k}, D_{[j-1]k+1} \right\} \qquad j = 2, 3, ..., n, \;\; k = 1, 2, ..., m-1; \\
&D_{[j]m} = D_{[j]m-1} + p_{[j]m} \qquad j = 1, 2, ..., n;
\end{aligned}
$$

Thus,

$$C_{\max} = D_{[n]m-1} + p_{[n]m}$$

5 Hybrid EDA for BFSP

In this section we present in detail an EDA to solve the Blocking Flowshop Scheduling Problem (BFSP), which is aimed at makespan minimization.

5.1 Solution Representation

For encoding the solution, we use the well-known representation scheme for the PFSP, that is the permutation of *n* jobs, where the *j*th number in the permutation denotes the job located in position *j*.

5.2 Initial Population

For generating the initial population of P individuals, we propose to generate P-1 individuals randomly and we apply NEH algorithm, proposed by Nawaz et al. (1983), for the remaining element.

NEH can be described as follows:

Step1: The jobs are sorted with respect to the decreasing order of sums of their processing times.
Step2: Take the first two jobs and evaluate the two possible schedules containing them. The sequence with better objective function value is taken for further consideration.
Step 3: Take every remaining job in the permutation given in Step 1 and find the best schedule, by placing it at all possible positions in the sequence of jobs that are already scheduled.

5.3 Selection

In our algorithm, we adopted the same procedure of selection employed by Reeves (1995) for solving the flowshop scheduling problem. We describe this procedure as follows.

First, for each individual p, the fitness value $f\left(p\right)=\dfrac{1}{C_{\max}\left(p\right)}$ is calculated, second the individuals of the initial population are sorted in ascending order according to their fitness, i.e. the individual with a higher makespan value will be at the top of the list. Finally, a set of Q individuals are selected from the sorted list.

5.4 Construction of a Probabilistic Model and Creation of New Individuals

The probabilistic model constitutes the main issue for an EDA and the performance of the algorithm is closely related to it (Lozano J.A et al., 2006), 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 select at random a sequence of jobs, denoted sr, from the set of 25% best solutions in the sorted list of sequences. Based on the priority rules of the order of the q first jobs in the s_r, we determine the estimation of distribution model while taking into account both the order of the jobs in the sequence and the similar blocks of jobs presented in the selected parents. In fact, the parameter q is an intensification parameter because, when it is possible, it leads to maintain the same structure of q first jobs and setting it to a constant value preserves the linearity of the algorithm.

Let:

– η_{jk} be the number of times of apparition 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 apparition of job *j* 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 employ these parameters in order to slow down the convergence of the algorithm.

– Let Ω_k be the set of *q* first jobs not already scheduled following their order in s_r until position *k*.

We define π_{jk} the probability of selection of the job *j* in the k^{th} position by the following formula:

$$\pi_{jk} = \frac{\eta_{jk} \times \mu_{j[k-1]}}{\sum_{l \in \Omega_k} \eta_{lk} \times \mu_{l[k-1]}}$$

For each position *k* in the sequence of a new individual, we select a job *j* among the set of *q* first jobs not already scheduled, following their order in s_r by sampling from the probability distribution π_{jk} .

5.5 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 available to decide 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 quits the population and is replaced with the new individual.

5.6 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. In our algorithm, we set a maximum number of generations and a maximal computational time.

5.7 Local Search

To improve the performance of EDA, the successful way is to hybridize it with local search methods (Lozano J.A. et al., 2006). We propose to apply a local search algorithm as an improvement procedure, after the creation of a new individual.

We propose to restrict the application of the local search procedure to a part of individuals by employing a probability of improvement that depends on the quality of the subjected individual. We define this probability as follows:

Let $p^c = \exp\left(\frac{RD}{\alpha}\right)$ be the calculated probability for application of local search, where:

$$RD = \left(\frac{f(x_{current}) - f(x_{best})}{f(x_{best})}\right)$$

with $x_{current}$ 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 the local search procedure to the individual under consideration.

At each iteration of the local search procedure, we select one among two kinds of neighbourhoods randomly. The first one leads to choose two distinct positions (i, j) at random, following the uniform distribution in the range $[1,n]$, and the jobs on these positions are exchanged. The second one consists in selecting at random a job j from the sequence and inserting it on a random position i. This procedure will be repeated as far as reaching the maximal number of iterations $iter_{max}$.

6 Computational Results

In this section, we discuss the performance of our proposed algorithms: EDA (without hybridization) and H-EDA. All computations for blocking flowshop scheduling problem, with respect to the makespan criterion, were implemented using C++ program and carried out on an Intel Pentium IV 3.2 GHz, RAM 512 MB based computer, running under Windows XP. In order to evaluate the performances of the proposed algorithms, the Taillard's instances were used for the flowshop scheduling problem (Taillard E., 1993). These instances consist of a set of 120 problems with sizes m=5, 10 and 20 and n=20, 50, 100, 200 and 500. The performance measure employed in our numerical study was average relative percentage deviation in makespan $\Delta_{average}$:

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

where Heu_i is the solution given by any of the R replications of the considered algorithms and $Best_{known}$ is the best solution provided by a competing algorithm for the specified problem or by one of our algorithms.

The parameters of the algorithms were fixed after a set of preliminary experiments, as follows: P = 60, $\delta_1 = \delta_2 = 4/n$, the number of the selected parents Q = 3, q = 20, the number of generated offspring $O = 3$. Numerically, $p^c = 0.5$ leads to accepting a sequence with a makespan superior by 5% relatively to the best value of makespan found.

So, $\beta = \dfrac{RD}{\log(p^c)} = \dfrac{0.01}{\log(0.5)}$ thereafter we determined p^c according to this formula:

$$p^c = \exp\left(\frac{RD}{\beta}\right).$$

The maximum number of iterations of the local search procedure was set to $2n^2$.

6.1 Comparison with GA

For testing the efficiency of our proposed EDA (without local search) against another evolutionary algorithm, we have implemented the GA of Caraffa et al. (2001). For performing a meaningful comparison we have set the same stopping criterion of 1000 generations for both algorithms.

The obtained results for each class of instances, over R=10 replications, are given in Table 1. For the small instances, with $n < 200$, in average, EDA outperforms GA both in terms of $\Delta_{average}$ and Δ_{max} , so, EDA can find better results than GA in average and worst case. Regarding Δ_{min} the two algorithms provide almost the same results. Also, for these instances, the range of changes for EDA solutions, i.e. the difference between Δ_{min} and Δ_{max}, is smaller than that range for GA, in average, thus EDA is more robust than GA. For large instances, with n = 200 and 500, EDA confirms its superiority, in terms of $\Delta_{average}$ and Δ_{max}, and it is better than GA for finding the best results (Δ_{min}). Although EDA is better than GA in term of solution quality, the latter appears faster after 1000 generations (Table 6).

Table 1. Comparison between EDA and GA

instances	EDA			GA		
	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}
20*05	0.01	0.02	0.03	0.01	0.03	0.05
20*10	0.01	0.02	0.03	0.01	0.03	0.05
20*20	0.01	0.01	0.02	0.01	0.02	0.03
50*05	0.02	0.03	0.04	0.03	0.04	0.06
50*10	0.02	0.03	0.04	0.02	0.04	0.06
50*20	0.03	0.03	0.04	0.01	0.03	0.05
100*05	0.04	0.05	0.06	0.05	0.06	0.07
100*10	0.03	0.04	0.04	0.03	0.04	0.06
100*20	0.02	0.03	0.03	0.02	0.03	0.04
200*10	0.04	0.04	0.05	0.06	0.07	0.08
200*20	0.03	0.03	0.03	0.04	0.04	0.05
500*20	0.02	0.02	0.02	0.05	0.05	0.05
average	0.02	0.03	0.03	0.03	0.04	0.05

Table 2. Results of H-EDA for 20 jobs instances

instances	Best known	RON	TS+M	H-EDA		
				Δ_{min}	Δ_{avg}	Δ_{max}
ta_20_5_01	**1374**	0.01	0.01	0.00	0.00	0.01
ta_20_5_02	1411	0.00	0.01	0.00	0.00	0.00
ta_20_5_03	**1280**	0.01	0.01	0.00	0.00	0.01
ta_20_5_04	1448	0.00	0.00	0.00	0.00	0.00
ta_20_5_05	**1342**	0.02	0.01	0.00	0.00	0.00
ta_20_5_06	1363	0.00	0.00	0.00	0.00	0.00
ta_20_5_07	1381	0.00	0.00	0.00	0.00	0.01
ta_20_5_08	**1379**	0.00	0.01	0.00	0.00	0.00
ta_20_5_09	**1373**	0.00	0.01	0.00	0.00	0.01
ta_20_5_10	1283	0.00	0.01	0.00	0.00	0.01
ta_20_10_01	1698	0.02	0.00	0.00	0.00	0.00
ta_20_10_02	**1833**	0.03	0.00	0.00	0.00	0.00
ta_20_10_03	**1659**	0.01	0.00	0.00	0.00	0.01
ta_20_10_04	**1535**	0.06	0.01	0.00	0.00	0.01
ta_20_10_05	**1617**	0.03	0.01	0.00	0.00	0.01
ta_20_10_06	**1592**	0.03	0.01	0.00	0.00	0.01
ta_20_10_07	**1622**	0.01	0.00	0.00	0.00	0.00
ta_20_10_08	**1731**	0.01	0.01	0.00	0.00	0.01
ta_20_10_09	**1747**	0.02	0.01	0.00	0.00	0.01
ta_20_10_10	1782	0.04	0.00	0.00	0.00	0.00
ta_20_20_01	**2436**	0.04	0.00	0.00	0.00	0.00
ta_20_20_02	**2234**	0.03	0.00	0.00	0.00	0.00
ta_20_20_03	**2480**	0.03	0.00	0.00	0.00	0.01
ta_20_20_04	2348	0.02	0.00	0.00	0.00	0.00
ta_20_20_05	**2435**	0.04	0.01	0.00	0.00	0.01
ta_20_20_06	**2389**	0.03	0.00	0.00	0.00	0.01
ta_20_20_07	2390	0.05	0.00	0.00	0.00	0.01
ta_20_20_08	**2328**	0.04	0.01	0.00	0.00	0.01
ta_20_20_09	**2363**	0.02	0.00	0.00	0.00	0.01
ta_20_20_10	**2323**	0.04	0.00	0.00	0.00	0.00
average		0.02	0.01	0.00	0.00	0.01

6.2 Performance of H-EDA

The performance of H-EDA is evaluated against the representative approaches developed for the same problem. The competing algorithms are the branch and bound algorithm of Ronconi (2005) and the Tabu Search of Grabowski and Pempera (2007), denoted by RON and TS+M respectively. We set the CPU time limit of each replication to $(n \times m) \times 20/3$ seconds.

Table 2 to Table 5 present the results found by our H-EDA. First, in total, our algorithm has improved 109 solutions out of 120 and, even for the 11 remaining instances,

Table 3. Results of H-EDA for 50 jobs instances

instances	**Best known**	**RON**	**TS+M**	**H-EDA**		
				Δ_{min}	Δ_{avg}	Δ_{max}
ta_50_5_01	**3055**	0.03	0.04	0.00	0.01	0.01
ta_50_5_02	**3249**	0.05	0.03	0.00	0.01	0.01
ta_50_5_03	**3056**	0.04	0.04	0.00	0.01	0.01
ta_50_5_04	**3170**	0.05	0.04	0.00	0.01	0.01
ta_50_5_05	**3200**	0.03	0.05	0.00	0.01	0.01
ta_50_5_06	**3224**	0.06	0.04	0.00	0.00	0.01
ta_50_5_07	**3079**	0.05	0.03	0.00	0.00	0.01
ta_50_5_08	**3097**	0.06	0.05	0.00	0.01	0.01
ta_50_5_09	**2963**	0.06	0.04	0.00	0.00	0.01
ta_50_5_10	**3160**	0.04	0.04	0.00	0.01	0.01
ta_50_10_01	**3737**	0.06	0.02	0.00	0.00	0.01
ta_50_10_02	**3562**	0.06	0.02	0.00	0.01	0.02
ta_50_10_03	**3554**	0.05	0.01	0.00	0.00	0.01
ta_50_10_04	**3754**	0.04	0.02	0.00	0.00	0.01
ta_50_10_05	**3698**	0.06	0.01	0.00	0.01	0.02
ta_50_10_06	**3678**	0.05	0.03	0.00	0.01	0.01
ta_50_10_07	**3765**	0.06	0.01	0.00	0.01	0.01
ta_50_10_08	**3632**	0.04	0.02	0.00	0.01	0.01
ta_50_10_09	**3604**	0.05	0.02	0.00	0.01	0.01
ta_50_10_10	**3691**	0.06	0.01	0.00	0.01	0.01
ta_50_20_01	**4591**	0.07	0.01	0.00	0.00	0.01
ta_50_20_02	**4373**	0.07	0.01	0.00	0.01	0.01
ta_50_20_03	**4354**	0.07	0.01	0.00	0.01	0.02
ta_50_20_04	**4448**	0.05	0.01	0.00	0.01	0.01
ta_50_20_05	**4353**	0.03	0.01	0.00	0.00	0.01
ta_50_20_06	**4368**	0.04	0.00	0.00	0.01	0.01
ta_50_20_07	**4386**	0.04	0.00	0.00	0.00	0.01
ta_50_20_08	**4415**	0.07	0.01	0.00	0.01	0.01
ta_50_20_09	**4400**	0.03	0.00	0.00	0.00	0.01
ta_50_20_10	**4502**	0.08	0.03	0.00	0.01	0.01
average		0.05	0.02	0.00	0.01	0.01

it can reach 9 upper bounds found by TS+M. Additionally, the most important improvement occurs for the instances with the size larger than 20. Especially when n = 50, 100 and 200, H-EDA has improved all upper bounds provided by previous approaches. In other hand, concerning the CPU time, in average, when we take into account the difference between the computer characteristics, H-EDA is faster than the TS+M approach (Table 6).

Table 4. Results of H-EDA for 100 jobs instances

instances	**Best known**	**RON**	**TS+M**	**H-EDA**		
				Δ_{min}	Δ_{avg}	Δ_{max}
ta_100_5_01	**6256**	0.01	0.06	0.00	0.00	0.01
ta_100_5_02	**6075**	0.00	0.06	0.00	0.01	0.01
ta_100_5_03	**6018**	0.01	0.05	0.00	0.00	0.01
ta_100_5_04	**5832**	0.00	0.05	0.00	0.01	0.02
ta_100_5_05	**6055**	0.02	0.06	0.00	0.00	0.01
ta_100_5_06	**5914**	0.00	0.05	0.00	0.01	0.02
ta_100_5_07	**6073**	0.00	0.05	0.00	0.00	0.01
ta_100_5_08	**5981**	0.00	0.06	0.00	0.00	0.01
ta_100_5_09	**6210**	0.00	0.06	0.00	0.01	0.01
ta_100_5_10	**6226**	0.00	0.05	0.00	0.01	0.01
ta_100_10_01	**7190**	0.02	0.02	0.00	0.00	0.01
ta_100_10_02	**6890**	0.03	0.04	0.00	0.01	0.01
ta_100_10_03	**7073**	0.01	0.03	0.00	0.00	0.01
ta_100_10_04	**7282**	0.06	0.02	0.00	0.01	0.02
ta_100_10_05	**6956**	0.03	0.03	0.00	0.01	0.02
ta_100_10_06	**6811**	0.03	0.03	0.00	0.00	0.01
ta_100_10_07	**6933**	0.01	0.03	0.00	0.01	0.01
ta_100_10_08	**6934**	0.01	0.02	0.00	0.02	0.02
ta_100_10_09	**7223**	0.02	0.02	0.00	0.00	0.01
ta_100_10_10	**7054**	0.04	0.03	0.00	0.01	0.02
ta_100_20_01	**8000**	0.04	0.01	0.00	0.01	0.01
ta_100_20_02	**8021**	0.03	0.02	0.00	0.00	0.01
ta_100_20_03	**8014**	0.03	0.01	0.00	0.01	0.01
ta_100_20_04	**8023**	0.02	0.01	0.00	0.01	0.01
ta_100_20_05	**8004**	0.04	0.01	0.00	0.01	0.01
ta_100_20_06	**8079**	0.03	0.02	0.00	0.00	0.01
ta_100_20_07	**8152**	0.05	0.02	0.00	0.01	0.01
ta_100_20_08	**8209**	0.04	0.02	0.00	0.00	0.01
ta_100_20_09	**8116**	0.02	0.01	0.00	0.00	0.01
ta_100_20_10	**8160**	0.04	0.01	0.00	0.01	0.01
average		0.02	0.03	0.00	0.01	0.01

Table 5. Results of H-EDA for 200 and 500 jobs instances

instances	Best known	RON	TS+M	H-EDA		
				Δ_{min}	Δ_{avg}	Δ_{max}
ta_200_10_01	**13718**	0.03	0.04	0.00	0.01	0.02
ta_200_10_02	**13618**	0.04	0.04	0.00	0.01	0.02
ta_200_10_03	**13779**	0.06	0.04	0.00	0.00	0.01
ta_200_10_04	**13718**	0.06	0.04	0.00	0.01	0.01
ta_200_10_05	**13763**	0.04	0.05	0.00	0.00	0.01
ta_200_10_06	**13472**	0.04	0.05	0.00	0.01	0.01
ta_200_10_07	**13869**	0.06	0.03	0.00	0.01	0.01
ta_200_10_08	**13848**	0.04	0.04	0.00	0.01	0.01
ta_200_10_09	**13580**	0.04	0.04	0.00	0.01	0.02
ta_200_10_10	**13712**	0.05	0.02	0.00	0.01	0.01
ta_200_20_01	**15122**	0.03	0.01	0.00	0.01	0.02
ta_200_20_02	**15379**	0.03	0.03	0.00	0.01	0.01
ta_200_20_03	**15528**	0.04	0.03	0.00	0.00	0.01
ta_200_20_04	**15331**	0.05	0.02	0.00	0.01	0.02
ta_200_20_05	**15295**	0.05	0.01	0.00	0.00	0.01
ta_200_20_06	**15387**	0.04	0.01	0.00	0.01	0.01
ta_200_20_07	**15370**	0.04	0.02	0.00	0.01	0.01
ta_200_20_08	**15386**	0.05	0.01	0.00	0.01	0.01
ta_200_20_09	**15279**	0.04	0.02	0.00	0.01	0.02
ta_200_20_10	**15375**	0.05	0.04	0.00	0.01	0.01
average		0.04	0.03	0.00	0.01	0.01
ta_500_20_01	**37530**	0.03	0.02	0.00	0.01	0.01
ta_500_20_02	**37942**	0.03	0.01	0.00	0.00	0.01
ta_500_20_03	**37637**	0.03	0.01	0.00	0.00	0.00
ta_500_20_04	**37888**	0.03	0.02	0.00	0.00	0.01
ta_500_20_05	**37622**	0.04	0.02	0.00	0.00	0.01
ta_500_20_06	**37950**	0.02	0.01	0.00	0.00	0.01
ta_500_20_07	37561	0.03	0.01	0.00	0.01	0.01
ta_500_20_08	37750	0.03	0.01	0.00	0.00	0.01
ta_500_20_09	**37521**	0.03	0.01	0.00	0.00	0.01
ta_500_20_10	**37869**	0.03	0.02	0.00	0.00	0.00
average		0.03	0.02	0.00	0.00	0.01

Table 6. Computational times

instances	EDA			GA			TS+M	H-EDA		
	min	average	max	min	average	max		min	average	max
20*05	0.24	0.87	1.43	0.00	0.02	0.08	2.70	0.03	0.28	0.60
20*10	0.21	0.93	1.57	0.00	0.02	0.08	4.60	0.10	0.67	1.24
20*20	0.32	0.95	1.74	0.01	0.05	0.17	7.60	0.13	1.37	2.43
50*05	2.54	4.14	4.99	0.08	0.16	0.27	6.20	0.19	0.85	1.55
50*10	2.34	4.35	5.30	0.11	0.26	0.46	10.80	0.45	1.74	2.95
50*20	2.89	4.61	5.76	0.22	0.51	0.92	19.30	0.53	3.14	5.98
100*05	6.51	9.50	10.94	0.48	0.57	0.61	12.40	0.77	1.97	3.28
100*10	8.11	10.22	11.59	0.81	1.03	1.12	22.10	1.49	3.90	6.44
100*20	6.34	10.26	12.41	1.47	1.88	2.04	39.40	3.00	8.20	12.88
200*10	17.76	22.26	24.38	1.99	2.15	2.22	44.30	9.54	12.40	13.34
200*20	18.39	24.18	26.77	3.74	4.05	4.23	79.40	18.10	24.22	26.67
500*20	48.20	70.65	82.86	10.30	10.71	11.78	209.00	66.67	66.67	66.67
average	9.49	13.58	15.81	1.60	1.78	2.00	38.15	8.42	10.45	12.00

7 Conclusion

In this chapter, we have proposed a hybrid EDA algorithm to minimize the makespan in the blocking flowshop scheduling problem. The probabilistic model built for our EDA depends on both the order of the jobs in the sequence and the similar blocks of jobs presented in the set of selected parents. A local search procedure is added to the EDA as an improvement phase, after creating a new individual. The computational results show that our proposed EDA, without hybridization, performs better than a GA previously developed for the same problem in terms of solution quality. However the GA outperforms our algorithm when 1000 generations are set as stopping criterion for both algorithms.

Also, by comparing the hybrid algorithm against competing approaches available in the literature, it's seen that our algorithm is better than these approaches, both in terms of solution's quality and computational times and it seems able to improve best known solutions.

References

Abadi, I.N.K., Hall, N.G., Sriskandarajah, C.: Minimizing cycle time in a blocking flowshop. Operations Research 48, 177–180 (2000)

Back, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford (1996)

Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, Technical Report, Carnegie Mellon Report, CMU-CS: 94-163 (1994)

Baluja, S., Davies, S.: Using optimal dependency trees for combinatorial optimization: Learning the structure of search space. Technical Report No. CMU-CS-97-107, Carnegie Mellon University, Pittsburgh, Pennsylvania (1997)

Caraffa, V., Ianes, S., Bagchi, T.P., Sriskandarajah, C.: Minimizing makespan in a blocking flowshop using genetic algorithms. International Journal of Production Economics 70, 101–115 (2001)

Companys, R., Mateo, M.: Different behaviour of a double branch-and-bound algorithm on Fm|prmu|Cmax and Fm|block|Cmax problems. Computers and Operations Research 34, 938–953 (2007)

DeBonet, J.S., Isbell, C.L., Viola, P.: MIMIC: Finding optima by estimating probability densities. In: Mozer, M., Jordan, M., Petsche, T. (eds.) Advances in Neural Information Processing Systems, vol. 9 (1997)

Fogel, D.B.: Evolutionary Computation. In: Toward a New Philosophy of Machine Intelligence. IEEE Press, Piscataway (1995)

Grabowski, J., Pempera, J.: Sequencing of jobs in some production system. European Journal of Operational Research 125, 535–550 (2000)

Grabowski, J., Pempera, J.: The permutation flow shop problem with blocking. A tabu search approach. OMEGA The International Journal of Management Science 35, 302–311 (2007)

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5, 287–326 (1979)

Hall, N.G., Sriskandarajah, C.: A survey of machine scheduling problems with blocking and no-wait in process. Operations Research 44, 510–525 (1996)

Harik, G., Lobo, F.G., Golberg, D.E.: The compact genetic algorithm. In: Proceedings of the IEEE Conference on Evolutionary Computation, pp. 523–528 (1998)

Li, H., Zhang, Q., Tsang, E., Ford, J.A.: Hybrid Estimation of Distribution Algorithm for Multi-objective Knapsack Problem. In: The 4th European Conference on Evolutionary Computation in Combinatorial Optimization, Coimbra, Portugal, 5-7 April (2004)

Larrañaga, P., Etxeberria, R., Lozano, J.A., Pena, J.M.: Combinatorial Optimization by learning and simulation of Bayesian networks. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford, pp. 343–352 (2000)

Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. In: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)

Leistein, R.: Flowshop sequencing with limited buffer storage. International Journal of Production Research 28, 2085–2100 (1990)

Levner, E.M.: Optimal Planning of Parts Machining on a Number of Machines. Automation and Remote Control 12, 1972–1978 (1969)

Lozano, J., Larraanaga, P., Inza, I., Bengoetxea, E.: Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms. Springer, Heidelberg (2006)

Lozano, J.A., Mendiburu, A.: EDAs applied to the job shop scheduling problem. In: Lozano, J.A., Larraanaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms, pp. 231–240. Springer, Heidelberg (2002)

McCormick, S.T., Pinedo, M.L., Shenker, S., Wolf, B.: Sequencing in an assembly line with blocking to minimize cycle time. Operations Research 37, 925–935 (1989)

Mühlenbein, H.: The equation for response to selection and its use for prediction. Evolut. Comput. 5, 303–346 (1998)

Mühlenbein, H., Mahnig, T.: The Factorized Distribution Algorithm for additively decomposed functions. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 752–759. IEEE press, Los Alamitos (1999)

Mühlenbein, H., Paaß, G.: From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. PPSN, 178–187 (1996)

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

Paul, T.K., Iba, H.: Linear and Combinatorial Optimizations by estimation of Distribution Algorithms. In: 9th MPS Symposium on Evolutionary Computation, IPSJ, Japan (2002)

Pelikan, M., Mühlenbein, H.: The bivariate marginal distribution algorithm. In: Roy, R., Furuhashi, T., Chandhory, P.K. (eds.) Advances in Soft Computing-Engineering Design and Manufacturing, pp. 521–535. Springer, Heidelberg (1999)

Pelikan, M., Goldberg, D.E., Cantpaz, E.: Linkage Problem, Distribution Estimation and Bayesian Networks. Evolutionary Computation 8(3), 311–340 (2000)

Reeves, C.R.: A genetic algorithm for flowshop sequencing. Computers and Operations Research 22, 5–13 (1995)

Ronconi, D.P.: A note on constructive heuristics for the flowshop problem with blocking. International Journal of Production Economics 87, 39–48 (2004)

Ronconi, D.P.: A branch-and-bound algorithm to minimize the makespan in a flowshop problem with blocking. Annals of Operations Research 138, 53–65 (2005)

Ronconi, D.P., Armentano, V.A.: Lower bounding schemes for flowshops with blocking in-process. Journal of the Operational Research Society 52, 1289–1297 (2001)

Salhi, A., Rodriguez, J.A.V., Zhang, Q.: An Estimation of Distribution Algorithm with Guided Mutation for a Complex Flow Shop Scheduling Problem GECCO 2007, London, England, United Kingdom, July 7–11 (2007)

Suhami, I., Mah, R.S.H.: An Implicit Enumeration Scheme for the Flowshop Problem with No Intermediate Storage. Computers and Chemical Engineering 5, 83–91 (1981)

Syswerda, G.: Simulated crossover in genetic algorithms. In: Foundations of Genetic Algorithms, vol. 2, pp. 239–255. Morgan Kaufmann, San Francisco (1993)

Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285 (1993)

Zhang, Q., Sun, J., Tsang, E.P.K., Ford, J.: Estimation of Distribution Algorithm with 2-opt Local Search for the Quadratic Assignment Problem. to be appeared in a book on Estimation of Distribution Algorithm. In: Lozano, J., Larraanaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms, pp. 281–291. Springer, Heidelberg (2006)

Zhang, Q., Zhou, A., Jin, Y.: RM-MEDA: A Regularity Model Based Multiobjective Estimation of Distribution Algorithm. IEEE Trans. Evolutionary Computation 12, 41–63 (2008)