Hybrid Estimation of Distribution Algorithm for permutation flowshop scheduling problem with sequence dependent family setup times

Mansour Eddaly[1], **Bassem Jarboui**[1], **Radhouan Bouabda**[1], **Abdelwaheb Rebai**[2]

[1]University of Sfax, FSEGS, route de l'aroport km 4, Sfax 3018, Tunisie (eddaly.mansour@gmail.com, bassem_jarboui@yahoo.fr, radhouan.bouabda@gmail.com)

[2]University of Sfax, ESC, route de l'aroport km 4, Sfax 3018, Tunisie (abdelwaheb.rebai@fsegs.rnu.tn)

ABSTRACT

This paper addresses to the scheduling in manufacturing cell environment with sequence dependent family setup times in a flow shop with respect to the makespan criterion. Since this is a NP-hard problem, we present an estimation of distribution algorithm as an evolutionary algorithm for solving it. In order to improve the quality of solution of our algorithm, we propose a hybridization with an iterated local search algorithm. The computational experiments show that our algorithm is better than the evolutionary algorithms proposed in the literature. Moreover, it seems able to provide good results in short computational time.

Keywords: manufacturing cell, flowshop, sequence dependant family, estimation of distribution algorithm, iterated local search.

1. Introduction

Since the early 1960's, the development of Cellular Manufacturing (CM), as one of Group Technology (GT) subsets, has received much attention of practitioners and researchers in the manufacturing systems. The major issues are the reduction of the complexity and the increase of the productivity of job shops [1]. Operationally, the CM consists in arranging a set of similar jobs into part families. These similarities concern the jobs setups, machinery requirements and operations. In [1] many industrial applications have been cited in CM environments such that chip producing, metal fabricating, and manual assembly industries. Hendizadeh et al. [2] have presented a detailed literature review of the scheduling of cellular manufacturing systems. In this paper we consider the scheduling of CM environment while processing the jobs with the same setup family together. Moreover, in each family, there is a set of n jobs that must be processed on a set of m in the same order without pre-emption (no interruption is allowed). Such cells are known as flowline cells or pure flowshop manufacturing cells [3]. This problem occurs when the time required to shift from one part (job) to another is negligible or included in the processing times. However, the time required to shift from one family to another is high and cannot be neglected. A comprehensive review can be found in [4] for the flowshop scheduling problems with setup times. They have included the problem under consideration in sequence dependent family setup times (SDFST) category. Schaller et al. [3] have presented a real world application in the manufacturing printing circuit boards where the use of CM technology is highly required. Garey and Johnson [5] have showed that this problem is strongly NP-hard with respect to the makespan criterion. This is justifying the choice of approximate algorithms more than exact methods for solving this problem. Schaller et al. [3] have developed lower bounds for solving this problem under minimizing the makespan. The obtained results have showed that the branch and bound based proposed lower bounds is efficient for the test problems with small sizes. However, when the number of families or the number of machines increases, the performance of the lower bounds decreases. Also, several constructive heuristics have been proposed in [3]. The computational results have showed that, for scheduling the jobs within families, the called C̈DSḧeuristic [6] is the best one. Besides, the NEH heuristic [7] is the most efficient regarding the scheduling job families. França et al. [8] have presented two evolutionary algorithms: a Genetic Algorithm (GA) and a Memetic Algorithm (MA). The computational results showed that these two algorithms outperform existing best heuristic. In addition MA presents a slight superiority according to the GA. Hendizadeh et al. [2] have proposed a Tabu Search (TS) algorithm (TS). They have employed the Simulated Annealing algorithm (SA) concepts for balancing between intensification and diversification of the search's direction. Their results have showed that the proposed algorithm, in average, outperforms the constructive heuristic of [3] in terms of solution quality but not in terms of CPU times. Moreover, the TS algorithm and the MA of Frana et al. [8] appear similar in average. Lin et al. [9] have proposed three different metaheureustics including SA, GA and TS for solving this problem. It's noted that the proposed SA and GA outperform MA of and Frana et al. [8] and TS of Hendizadeh et al. [2] both regarding the solution quality and the CPU times.

Recently, a novel evolutionary technique, called Estimation of Distribution Algorithm (EDA) has been proposed by Mŭhlenbein and Paaβ [10]. For generating a new individual, EDA uses a probabilistic model learned from a population of individuals. Therefore, a distribution of probability is estimated from the selected candidates from the initial population and then new offspring is generated according to this distribution.

In this paper an EDA is developed for solving the flowshop scheduling problem with sequence dependent family setup times. Moreover, an Iterated Local Search algorithm is probabilistically added to the EDA aiming to enhance the quality of the solution. The remainder of this paper is organized as follows: section 2 presents the mathematical formulation of the problem; section 3 presents the proposed EDA. The computational results are presented in section 4. Finally, the conclusion is given in section 5.

978-1-4244-4136-5/09/$25.00©2009IEEE

2. Problem Formulation

In a flowshop scheduling problem with sequence dependent family setup times, it's assumed that in each family f $(f = 1, 2, ..., F)$ there is a set of n_f jobs must to be processed on a set of m machines where the processing sequence of the jobs is the same for all machines. In particular, we suppose that the job setup times within each family are included in the job processing times. However, if a family f follows another f' immediately in the family sequence, the time required for switching from f' to f on machine i $(i = 1, 2, ..., m)$ is denoted by $s_{[f][f']i}$. Also, we denote by $D_{f,j,i}$ and $p_{f,j,i}$ the departure time (starting time) and the processing time of the job j $(j = 1, 2, ..., n_f)$ on the machine i in the family f successively. Similarly, we denote by $D_{[f][j]i}$and $p_{[f][j]i}$ the departure time and the processing time of the job at the j^{th} position in the job sequence on the machine i in the f^{th} position in the family sequence successively. The makespan (C_{max}) can be found through the recursive expression according to the departure times as follows:

$D_{[1][1]1} = s_{[1][1]1}$
$D_{[1][1]i} = max\{s_{[1][1]i}, D_{[1][1]i-1} + p_{[1][1]i-1}\}$ $i = 2, 3, ..., m$
$D_{[1][j]1} = D_{[1][j-1]1} + p_{[1][j-1]1}$ $j = 2, 3, ..., n_1$
$D_{[1][j]i} = max\{D_{[1][j-1]i} + p_{[1][j-1]i}, D_{[1][j-1]i} + p_{[1][j]i-1}\}$ $j = 2, 3, ..., n_1$ $i = 2, 3, ..., m$
$D_{[f][1]1} = D_{[f-1][n_{f-1}]1} + p_{[f-1][n_{f-1}]1} + s_{[f][f-1]1}$ $f = 2, 3, ..., F$
$D_{[f][j]1} = D_{[f][j-1]1} + p_{[f][j-1]1}$ $f = 2, 3, ..., F$ $j = 2, 3, ..., n_f$
$D_{[f][1]i} = max\{D_{[f][n_{f-1}]i} + p_{[f][n_{f-1}]i} + s_{[f][f-1]i}, D_{[f][1]i-1} + p_{[f][1]i-1}\}$ $f = 2, 3, ..., F$ $i = 2, 3, ..., m$
$D_{[f][j]i} = max\{D_{[f][j-1]i} + p_{[f][j-1]i}, D_{[f][j]i-1} + p_{[f][j]i-1}\}$ $f = 2, 3, ..., F$ $j = 2, 3, ..., n_f$ $i = 2, 3, ..., m$
Thus
$C_{max} = D_{[F][n_f]m} + p_{[F][n_f]m}$

3. The proposed EDA

The section discusses the framework of our proposed EDA for solving the flowshop scheduling problem with sequence dependent family setup times under the minimization of the makespan criterion.

3.1. Encoding scheme and initial population

In this chapter each solution is represented by two permutations. The first permutation represents the family vectors where the f^th case indicates the family located on the position f in the family sequence. The second one represents the permutation of jobs within each family, where the j^th number denotes the job located on position j. In order to grant the diversification of the algorithm, the P individuals of the initial population are generated randomly.

3.2. Selection

The procedure of selection adopted in our algorithm consists of two phases. First, the individuals of the initial population are sorted, in an increasing order, according to their objective functions. Second, M individuals are selected from the subset of 20% of best individuals from the sorted list.

3.3. Estimation of the probability

In order to generate new individual, the EDA constructs a probabilistic model based on the selected individuals. We propose to build a probabilistic model for generating both the family sequence and the job sequence. As in [11] our estimated probabilities depend on the structure of sequences of selected individuals. Therefore, both the order of the family (job) and the similar blocks of families (jobs) are greatly considered. First we create the new family sequence by using the following parameters:
- η_{fg} be the number of times where the family f is located before or on the position g in the subset of the selected sequences plus a constant δ_1^f . Thus, η_{fg} indicates the importance of the order of the families in the family sequence.
- $\mu_{f[g-1]}$ be the number of times where the family f come after the family on the position $f' - 1$ in the subset of the selected sequences plus δ_2^f. $\mu_{f[g-1]}$ highlights the importance of the similar blocks of families in the family sequences. In such way, we prefer to conserve the similar blocks as much as possible.
- Ω_g^f is the set of families not already scheduled until position g.
Thus, the probability for locating the family f on the position g in the sequence of new individual is calculated as follows:

$$\pi_{fg} = \frac{\eta_{fg} \times \mu_{f[g-1]}}{\sum_{l \in \Omega_g} \eta_{lg} \times \mu_{l[g-1]}}$$

Then, given the resulted family sequence, we create the new job sequences by a similar way. Therefore, we redefine the same parameters as above (η_{jk},δ_1^j ,$\mu_{j[k-1]}$,δ_2^j and Ω_k^j) while modifying family sequence by job sequence. So, within each family f, the estimated probability of selection the job j on the k^th position in the new job sequence is obtained following this formula:

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

3.4. Iterated Local Search Algorithm

In order to improve the solution provided by the EDA, we propose to use an Iterated Local Search (ILS) algorithm [12]. This algorithm is a local search based algorithm characterised by its simplicity of implementation. Its previous application to hard combinatorial optimization problems, such as quadratic assignment problem [13], graph coloring [14], vehicle routing problem [15] and permutation flowshop scheduling problem [16], are qualified as successful. For each created individual, we calculate the probability which decides if the ILS algorithm will be applied this individual. This probability was proposed by Jarboui et al. [11], it depends on the quality of the new solution.

Let $\sigma_{current}$ and $f(\sigma_{current})$ denote the sequence of the new individual and its makespan respectively. Also, we denote by σ_{best} and $f(\sigma_{best})$ the best solution found by the algorithm and its objective function value. The probability of application of the ILS algorithm is defined as follows: $prob = max[exp(\frac{RD}{\alpha}), \epsilon]$ where $RD = (\frac{f(\sigma_{current})-f(\sigma_{best})}{f(\sigma_{best})})$. The basic framework an ILS algorithm consists of two main components: the local search procedures and the perturbation.
In our algorithm we use two local procedures for the family sequence as well as for the job sequence. We note that each procedure is applied to the family sequence and then for the job sequence within each family successively. The first procedure consists in inserting all pairs of families (jobs) in all possible positions in $\sigma_{current}$. We denote by σ_1 the obtained local optima from the insert moves. Next,σ_2 is obtained by the second local search after permuting all pairs of families (jobs) in $\sigma_{current}$. If σ_2 is better than σ_1 then the latter is replaced by the former and we return to the first procedure. These procedures continue to perform until no possible improvement.
As in the local search phase, the same way of perturbation is employed to both the family and the job sequence. Therefore, at each iteration of the ILS algorithm, two distinct positions are selected at random from the family (job) sequence and the families on these positions are exchanged. The ILS algorithm will terminate after a maximal number of iterations.

Tab. 2: Computational results of evolutionary algorithms

	MA				GA				EDA			
instances	Δ_{min}	Δ_{avg}	Δ_{max}	t_{avg}	Δ_{min}	Δ_{avg}	Δ_{max}	t_{avg}	Δ_{min}	Δ_{avg}	Δ_{max}	t_{avg}
LSU33	0.00	0.07	1.12	3.1	0.00	0.08	1.12	2.56	0,00	0,08	1,13	0,01
LSU34	0.00	0.32	2.43	10.1	0.00	0.33	2.38	2.69	0,00	0,33	2,44	0,03
LSU44	0.00	0.20	1.09	10.1	0.00	0.20	1.09	5.29	0,00	0,20	1,10	0,06
LSU55	0.00	0.28	1.86	12.1	0.00	0.28	1.83	8.01	0,00	0,28	1,86	0,22
LSU56	0.00	0.51	2.42	21.1	0.00	0.50	2.37	8.21	0,00	0,51	2,43	0,21
LSU65	0.00	0.31	2.42	15.1	0.00	0.32	2.37	10.97	0,00	0,31	2,43	0,28
LSU66	0.00	0.19	1.36	15.3	0.00	0.21	1.35	12.90	0,00	0,20	1,37	0,33
LSU88	0.00	0.58	1.86	24.7	0.00	0.58	1.83	24.59	0,00	0,59	1,86	1,18
LSU108	0.00	0.47	1.19	29.8	0.00	0.43	1.12	39.59	0,00	0,42	1,01	3,29
LSU1010	0.00	0.77	2.27	29.5	0.00	0.62	1.43	43.61	0,00	0,57	1,44	5,98
MSU33	0.00	0.37	3.37	7.1	0.00	0.35	3.26	2.54	0,00	0,36	3,37	0,01
MSU34	0.00	0.56	2.29	13.1	0.00	0.56	2.25	2.69	0,00	0,57	2,30	0,03
MSU44	0.00	0.50	2.32	19.1	0.00	0.50	2.27	5.33	0,00	0,50	2,33	0,08
MSU55	0.00	0.45	2.09	15.1	0.00	0.46	2.05	7.58	0,00	0,45	2,10	0,31
MSU56	0.00	0.87	3.12	24.1	0.00	0.89	3.03	8.23	0,00	0,87	3,13	0,34
MSU65	0.00	0.36	1.22	16.2	0.00	0.37	1.21	10.79	0,00	0,37	1,23	0,43
MSU66	0.00	0.50	1.63	22.7	0.00	0.50	1.61	12.78	0,00	0,51	1,63	0,53
MSU88	0.00	0.99	2.98	27.3	0.00	0.99	2.89	23.83	0,00	1,00	2,98	1,38
MSU108	0.00	0.86	1.80	30.1	0.00	0.82	2.24	39.66	0,00	0,78	1,63	6,86
MSU1010	0.15	1.15	2.53	30.8	0.15	0.98	2.82	43.71	0,15	0,97	2,42	5,53
SSU33	0.00	0.31	2.47	7.1	0.00	0.31	2.42	2.55	0,00	0,31	2,48	0,07
SSU34	0.00	0.83	2.94	15.1	0.00	0.82	2.86	2.71	0,00	0,83	2,94	0,07
SSU44	0.00	0.57	2.90	15.0	0.00	0.62	2.82	5.32	0,00	0,58	2,91	0,10
SSU55	0.00	0.92	2.34	26.0	0.00	0.93	2.29	7.57	0,00	0,92	2,35	0,56
SSU56	0.00	1.56	3.08	27.4	0.00	1.55	3.00	8.25	0,00	1,57	3,09	0,61
SSU65	0.00	0.99	3.44	24.0	0.00	1.00	3.33	10.86	0,00	1,24	2,73	1,89
SSU66	0.00	1.28	2.72	29.1	0.00	1.31	2.65	12.82	0,00	0,99	3,45	1,17
SSU88	0.28	1.85	3.31	30.3	0.29	1.88	3.34	23.82	0,29	1,79	3,46	2,97
SSU108	0.72	1.77	2.9	30.7	0.72	1.51	2.83	39.65	0,48	1,49	2,71	8,50
SSU1010	0.59	2.33	3.65	30.6	0.59	2.18	3.33	43.75	0,59	2,08	3,42	12,93
Average	**0.06**	**0.76**	**2.37**	**20.4**	**0.06**	**0.74**	**2.31**	**15.76**	**0.05**	**0.72**	**2.32**	**1.87**

MA was run on a PC Pentium II 266 MHz processor and 128 Mb RAM. (França et al.,2005)
GA was run on a PC Pentium IV 2.4GHz processor and 512Mb RAM.(Lin et al.,2009)

4. Computational experiments

Tab. 1: Computational results of local search based algorithms

	TS(2)				SA				TS(1)			
instances	Δ_{min}	Δ_{avg}	Δ_{max}	t_{avg}	Δ_{min}	Δ_{avg}	Δ_{max}	t_{avg}	Δ_{min}	Δ_{avg}	Δ_{max}	t_{avg}
LSU33	0.00	0.08	1.13	3.3	0.00	0.08	1.12	1.59	0.00	0.08	1.12	1.05
LSU34	0.00	0.33	2.44	10.2	0.00	0.33	2.38	1.79	0.00	0.33	2.38	1.31
LSU44	0.00	0.20	1.10	11.9	0.00	0.20	1.09	3.20	0.00	0.20	1.09	2.51
LSU55	0.00	0.29	1.86	13.0	0.00	0.28	1.83	4.75	0.00	0.28	1.83	4.03
LSU56	0.00	0.52	2.43	21.5	0.00	0.50	2.37	5.47	0.00	0.50	2.37	4.80
LSU65	0.00	0.31	2.43	16.5	0.00	0.31	2.37	6.67	0.00	0.31	2.37	5.78
LSU66	0.00	0.23	1.37	19.4	0.00	0.20	1.35	8.33	0.00	0.23	1.35	7.48
LSU88	0.00	0.65	2.05	25.4	0.00	0.58	1.98	16.64	0.00	0.68	1.98	15.70
LSU108	0.11	0.48	0.97	31.7	0.00	0.40	0.77	27.79	0.12	0.61	1.25	27.99
LSU1010	0.00	0.69	1.80	31.2	0.00	0.58	1.42	32.49	0.11	0.92	1.77	32.28
MSU33	0.00	0.38	3.37	7.3	0.00	0.35	3.26	2.59	0.00	0.35	3.26	2.11
MSU34	0.00	0.59	2.30	14.9	0.00	0.56	2.25	1.79	0.00	0.56	2.25	1.34
MSU44	0.00	0.58	2.51	20.4	0.00	0.50	2.27	3.23	0.00	0.50	2.27	2.51
MSU55	0.00	0.50	2.10	17.1	0.00	0.45	2.05	4.83	0.00	0.45	2.05	4.01
MSU56	0.00	0.90	3.13	25.1	0.00	0.86	3.03	5.53	0.00	0.88	3.03	4.77
MSU65	0.00	0.40	1.72	17.9	0.00	0.37	1.21	6.76	0.00	0.44	1.45	5.76
MSU66	0.00	0.52	1.63	23.8	0.00	0.50	1.61	8.34	0.00	0.53	1.61	7.46
MSU88	0.00	1.10	2.98	28.8	0.00	0.96	2.89	16.55	0.00	1.15	3.09	15.86
MSU108	0.00	1.17	3.05	31.3	0.00	0.78	1.77	27.79	0.16	1.30	2.55	27.26
MSU1010	0.15	1.22	3.71	31.9	0.15	0.98	2.36	32.64	0.19	1.41	3.73	32.33
SSU33	0.00	0.31	2.48	8.8	0.00	0.31	2.42	2.59	0.00	0.31	2.42	2.12
SSU34	0.00	0.96	6.62	16.9	0.00	0.82	2.86	1.79	0.00	0.82	2.86	1.41
SSU44	0.00	0.64	2.91	17.0	0.00	0.57	2.82	3.19	0.00	0.60	2.82	2.54
SSU55	0.00	0.94	2.35	26.8	0.00	0.90	2.29	4.75	0.00	0.94	2.29	4.03
SSU56	0.00	1.63	3.09	28.9	0.00	1.53	3.00	5.47	0.00	1.61	3.00	4.79
SSU65	0.00	1.03	3.45	24.6	0.00	0.96	3.33	6.78	0.00	1.02	3.33	5.82
SSU66	0.00	1.34	2.73	30.2	0.00	1.23	2.65	8.35	0.00	1.36	2.65	7.51
SSU88	0.29	2.14	4.53	30.9	0.29	1.76	3.46	16.66	0.29	2.12	4.12	15.83
SSU108	0.72	1.97	3.13	31.9	0.48	1.53	2.64	27.86	0.86	2.02	2.98	27.30
SSU1010	0.68	2.69	4.60	32.4	0.59	2.14	3.53	32.68	0.68	2.86	4.16	32.59
Average	**0.06**	**0.83**	**2.67**	**21.7**	**0.05**	**0.72**	**2.28**	**10.96**	**0.08**	**0.85**	**2.45**	**10.34**

TS(2) was run on a PC Pentium IV 2.6GHz processor and 512Mb RAM. (Hendizadeh et al.,2005)
SA and TS(1) were run on a PC Pentium IV 2.4GHz processor and 512Mb RAM. (Lin et al.,2009)

Several tests were carried out to evaluate the effectiveness and the efficiency of our EDA. In our implementation, we use the test problems tested by Schaller et al. [3], França et al. [8] and Lin et al. [9] for the same problem. In total, there are 900 instance problems classified into three classes according to the range of setup times named " Small Setups " (SSU), " Medium Setups " (MSU) and " Large Setups " (LSU). A Detailed description of these problem tests is given in [3] and França et al. [8]. Algorithm EDA was coded in C++ and run on a PC with Pentium 4, 3 GHz processor, with 512 Mb of RAM.
In order to properly set the values of tuning parameters, EDA was run several times. Therefore, we set P=60 (the size of initial population), $M = 3$ (the number of selected individuals),$\delta_1^f = \delta_2^f = \frac{4}{F}$,$\delta_j^j = \delta_2^j = \frac{4}{n_f}$. The parameter α used in the probability of application of the ILS algorithm is fixed according to RD and $prob$. Assuming that with a $prob = 0.5$, we accept a sequence with a C_{max} higher than the best value while RD is less than or equal to 1%. So,$\alpha = \frac{RD}{log(prob)} = \frac{0.01}{log(0.5)}$, thereafter we determined $prob$ according to this formula $prob = max[exp(\frac{RD}{\alpha}), \epsilon]$ with $\epsilon = 0.01$. Finally, we set a run time of 30 seconds as a termination criterion of our algorithm.
For each class of instances, we define a performance measure of each algorithm by the average relative percentage of improvements $\Delta_{average} = \frac{\sum_{i=1}^{30}(\frac{Heu_i - LB_i}{LB_i})}{30}$ of the makespan obtained by the heuristic algorithm i with respect to the lower bound values of Schaller et al. [3]. Δ_{min} and Δ_{max} denote, respectively, the minimum and the maximum relative percentage of improvements over each class of instances. tavg reports the average CPU times of different algorithms in seconds.
The results given in Table 1 are those obtained by the local search based approaches in the literature. The first column provides the results of The TS algorithm (TS(2)) of Hendizadeh et al. [2]. The second column and the third column show, respectively, the results of Simulated Annealing (SA) and TS algorithm (TS(1)) developed by Lin et al. [9]. Table 2 summarizes the results obtained by evolutionary algorithms including the Memetic Algorithm (MA) of Frana et al. [8], GA of Lin et al. [9] and our proposed EDA algorithms.
From Table 2, the values of Δ_{min}, $\Delta_{average}$ and Δ_{max}

provided by our algorithm are, respectively, equal to 0.05, 0.72 and 2.32. Therefore, while comparing our algorithm with the other evolutionary algorithms of the literature such as GA and MA, it's clear that, in average, our algorithm outperforms these algorithms in terms of quality of solutions. Concerning the computational effort, with respect to our used configuration, we find that our t_{avg} is equal to 1.87 seconds. So, our proposed EDA appears able to provide a good quality of solution in a short time.
Besides, the comparison against the local search based algorithms of the literature shows that the EDA is better than the two TS algorithms while regarding Δ_{min}, Δ_{avg} and Δ_{max}. However EDA and SA are almost similar with a slight superiority in favour of the latter in terms of Δ_{max}. On other hand, for the instances with large number of families and machines like SSU1010 and MSU1010, we observe that the EDA algorithm is better than SA in average.

5. Conclusion and further direction

This paper considered the flowline manufacturing cell scheduling problem with sequence dependent family setup times under the makespan minimization. Due to its hard complexity, there is a considerable amount of approximate methods that addressed this problem. We proposed an estimation of distribution algorithm that exploits the characteristics of order and blocks of a sequencing problem in the step of construction of new individual. An iterated local algorithm is embedded to the EDA for improving the quality of generated offspring. Numerical experiments indicated that our proposed algorithm is better than the evolutionary algorithm proposed in the literature. In addition, our EDA is able to reach good solutions with low computation effort. Further, this algorithm may be applied to solve other variants of the problem by relaxing the constraint of permutation schedules and/or the constraint of dependencies between the families.

REFERENCES

[1] Greene TJ, Sadowski RP. “A review of cellular manufacturing assumptions, advantages and design techniques”. *Journal of Operations Management*,Vol.4, pp85-97, 1984.

[2] Hendizadeh SH, Faramarzi H, Mansouri SA, Gupta JND, ElMekkawy TY. “Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times”. *International Journal of Production Economics*, Vol. 111, pp593-605, 2008.

[3] Schaller JE, Gupta JND, Vakharia AJ. “Scheduling a flowline manufacturing cell with sequence dependent family setup times”. *European Journal of Operational Research*, Vol. 125, pp324-39, 2000.

[4] Cheng TCE, Gupta JND, Wang G. “A review of flowshop scheduling research with setup times”. *Production and Operations Management*, Vol. 9, pp262-282, 2000.

[5] Garey MR, Johnson DS. “Computers and Intractability: A Guide to the Theory of NP-Completeness”. *Freeman, San Francisco, CA* 1979.

[6] Campbell, HG, Dudek RA, Smith ML. “A heuristic algorithm for the n-job, m-machine sequencing problem”. *Management Science*, Vol. 16, pp630-637, 1970.

[7] Nawaz M., Enscore E., Ham I. “A heuristic algorithm for the m-machine, n-job flow shop sequencing problem”. *OMEGA, International Journal of Management Science*, Vol. 11, pp91-95, 1983.

[8] França PM, Gupta JND, Mendes AS, Moscato P, Veltink K. “Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups”. *Computers and Industrial Engineering*, Vol. 48, pp491-506, 2005.

[9] Lin S-W, Ying K-C, Lee Z-J. “Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times”. *Computers and Operations Research*, Vol. 36, pp1110-1121, 2009.

[10] Mühlenbein H, Paaβ G., “From Recombination of Genes to the Estimation of Distributions I. Binary Parameters”. *PPSN* pp178-187, 1996.

[11] Jarboui B., Eddaly M., Siarry P., “An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems”. *Computers and Operations Research*, Vol. 36, pp2638-2646, 2008.

[12] Lourenc HR, Martin O, Stützle T. “Iterated local search”, in: F. Glover, G. Kochenberger (Eds.), *Handbook of Metaheuristics, International Series in Operations Research and Management Science*, Vol. 57, Kluwer Academic Publishers, Norwell, MA, pp321-353, 2002.

[13] Stützle T. “Iterated local search for the quadratic assignment problem”. *European Journal of Operational Research* Vol. 174, pp1519-1539, 2006.

[14] Caramia M, Dell’Olmo P. “Coloring graphs by iterated local search traversing feasible and infeasible solutions” *Discrete Applied Mathematics*, Vol. 156, pp201-217, 2008.

[15] Ibarakia T, Imahorib S Nonobec K, Sobued K, Unoe T, Yagiuraf M. “An iterated local search algorithm for the vehicle routing problem with convex time penalty functions”. *Discrete Applied Mathematics*, Vol. 156, pp2050-2069, 2008.

[16] Dong X, Huang H, Chen P. “An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion” *Computers and Operations Research*, Vol. 36, pp1664-1669, 2009.