Hybrid Flowshop with Unrelated Machines, Sequence Dependent Setup Time and Availability Constraints: An Enhanced Crossover Operator for a Genetic Algorithm

Victor Yaurima[1], Larisa Burtseva[2], and Andrei Tchernykh[3]

[1] CESUES Superior Studies Center, San Luis R.C., Mexico
yaurima@iing.mxl.uabc.mx
[2] Autonomous University of Baja California, Mexicali, Mexico
lpb@iing.mxl.uabc.mx
[3] CICESE Research Center, Ensenada, Mexico
chernykh@cicese.mx

Abstract. This paper presents a genetic algorithm for a scheduling problem frequent in printed circuit board manufacturing: a hybrid flowshop with unrelated machines, sequence dependent setup time and machine availability constraints. The proposed genetic algorithm is a modified version of previously proposed genetic algorithms for the same problem. Experimental results show the advantages of using new crossover operator. Furthermore, statistical tests confirm the superiority of the proposed variant over the state-of-the-art heuristics.

1 Introduction

In this paper, a scheduling problem for a hybrid flowshop (HFS) is considered. The HFS problem is a general case of the simple flowshop problem [1]. In simple flow shop problems, each machine operation center includes just one machine. In a HFS for at least one machine center or stage, there exists more than one machine available for processing. The HFS differs from the flexible flow line [2] and the flexible flowshop [3] problems, although many authors do not distinguish these three terms. In a flexible flow line as well as in a flexible flowshop the machines available at each stage are identical. A HFS does not have this restriction [4, 5, 6, 7, 8].

This paper deals with the HFS scheduling problem with unrelated parallel machines at the stages, sequence dependent setup time and availability constraints of the machines.

HFS has been introduced in 1971 by Arthanary and Ramaswamy [9], where a branch and bound algorithm (B&B) for a flexible flow shop with only two stages was proposed. One of the earliest works that deals with problems with m stages is presented by Salvador [10] in 1973, where dynamic programming algorithms for the no-wait flow shop with multiple processors were proposed.

R. Wyrzykowski et al. (Eds.): PPAM 2007, LNCS 4967, pp. 608–617, 2008.

Ruiz and Maroto [11], and Vignier et al. [12] present comprehensive surveys on the HFS. Many papers deal with simplified cases of the HFS with only two and three stages. Six studies about unrelated parallel machines with multiples stages [11,13,14,15,16,17] are known. Three of them take into account sequence dependent setup times, and only [11] considers availability constraints.

Adler et al. [17] have developed BPSS (Bagpak Production Scheduling System). The production problem contains setup times and, in some stages, unrelated parallel machines. The system is specific for the considered problem, and is based on the application of priority rules. Aghezzaf et al. [14] have proposed several methods to solve a problem in the carpet manufacturing industry where three stages and sequence dependent setup times are considered. The solution is based on the problem decomposition, heuristics, and mixed-integer programming models. Gourgand et al. [16] have presented several Simulated Annealing (SA)-based algorithms for the HFS problem. A specific neighborhood is used. The methods were applied to a real industrial problem. Allaoui and Artiba [13] have dealt with the HFS scheduling problem under maintenance constraints to optimize several objectives based on flow time and due date. In this model, the authors take also into consideration setup, cleaning and transportation times. The HFS scheduling problem with machine setup time dependent on job sequences, but without the availability constraints is considered in the paper of Lixin Tang and Yanyan Zhang [18]. In the model, all jobs pass the same route in the HFS, there is at least one identical machine at each stage, and at least one stage has more than two machines. A modification of the traditional Hopfield network formulation and the improving strategy are proposed. Zandieh at al. [15] have proposed an immune algorithm for considered problem. In the model, identical machines on each stage are considered. Obtained results are compared with those by Random Key Genetic Algorithm (RKGA) [19]. It was shown that the immune algorithm outperforms RKGA. Ruiz and Maroto [11] have developed a genetic algorithm to a complex generalized flowshop scheduling problem that consists of unrelated parallel machines at each stage, sequence dependent setup times and availability constraints applied to the production of textiles and ceramic tiles. It was shown that the proposed algorithm is more effective than all other. To the best of our knowledge, only latter method has been proposed for the HFS with unrelated parallel machines, sequence dependent setup time and availability constraints.

In this paper, a genetic algorithm to deal with the HFS problem with unrelated parallel machines, sequence dependent setup time, and machine availability constraints is introduced. Makespan (C_{max}) minimization is considered as the optimization criterion. This algorithm is a variant of the genetic algorithm for the HFS proposed by Ruiz and Maroto [11]. The restart conditions, crossover operations and stopping criterion are modified. It is shown that the variant achieves better results.

The rest of the paper is organized as follows. In Section 2, the problem statement is given. In Section 3, the proposed genetic algorithm variant is introduced. The experimental setup and evaluation results are presented in Section 4. Finally, Section 5 summarizes the paper and points out ideas for future research.

2 Problem Statements

Let a set N of n jobs, $N = \{1, ..., n\}$, has to be processed on a set M of m stages, $M = \{1, ..., m\}$. At every stage $i \epsilon M$, a set $M_i = \{1, ..., m_i\}$ of unrelated parallel machines that can process the jobs is given, where $|M_i| \geq 1$. Every job has to pass through all stages, and must be processed by exactly one machine at every stage. Let $p_{i_l,j}$be the processing time of the job $j \epsilon N$ on machine $l \epsilon M_i$ at the stage i. A machine based sequence dependent setup time is also considered. Let $S_{i_l,j,k}$ be the setup time in the machine l at the stage i when processing job $k \epsilon N$, after processing job j. For the stage i, E_{ij} is a set of eligible machines that can process job j, $1 \leq |E_{ij}| \leq m_i$.

Gourgand et al. [16] showed that for the given problem the total number of possible solutions is $n! \left(\prod_{i=1}^{m} m_i\right)^n$. Moreover, Gupta [6] showed that the flexible flowshop problem with only two stages ($m = 2$) is NP-hard even when one of the two stages contains a single machine. Since HFS is a general case of the flexible flowshop, we can conclude that HFS is also NP-hard. Using the well-known three field notation $\alpha \,|\beta|\, \gamma$ for scheduling problems and its extension for HFS proposed by Vignier et al. [12], the problem considered here can be viewed as $FHm, \left(\left(RM^{(i)}\right)_{i=1}^{(m)}\right) |S_{sd}, M_j|\, C_{max}$. The calculation of the C_{max} is as follow:

$$C_{i,\pi_{(j)}} = min_{l=1}^{m_j}\{max\{C_{i,L_{i_l}} + S_{i_l,L_{i_l},\pi_{(j)}}; C_{i-1,\pi_{(j)}}\} + p_{i_l,\pi_{(j)}}\} \tag{1}$$

where: π is permutation or sequence; $\pi_{(j)}$ is the job in the jth position in the sequence, $j \epsilon N$. Every job has to be processed at every stage, so m tasks per job are considered. $C_{i,\pi_{(j)}}$ is the completion time of job at stage i, where $i \epsilon M$. L_{i_l} is the last job that was assigned to machine l within stage i, $l \epsilon M_i$. $S_{i_l,L_{i_l},\pi_{(j)}}$ represents the setup time of machine l at stage i when processing job $\pi_{(j)}$ after having processed the previous work assigned to this machine $l(L_{i_l})$. Once all jobs are assigned to machines at all stages the makespan is calculated as follow:

$$C_{max} = max_{j=1}^{n}\{C_{m,\pi_{(j)}}\} \tag{2}$$

3 Genetic Algorithm

3.1 Encoding

The sequence of jobs represents an individual, which is called chromosome. In this encoding, a string of n integers, which is a permutation of the set $\{1, 2, ..., n\}$ is used, where each integer represents a job number. The following is the GA_{BC} procedure.

```
Input: The population of Psize individuals.
Output: A candidate sequence of length n.
01  Generate population
02  while not stopping_criterion do
03     for i=0 to Psize
```

```
04        generate_individual(i)
05        evaluate_objective_function(i)
06     select individuals by the tournament selection
07     keep the best individual found
08     if minimum_makespan_has_not_changed = 25 then
09        regenerate_population
10        if regenerate = 10 then
11           stopping_criterion = true
12        else increase regenerate
13        minimum_makespan_has_not_changed = 0
14     else
15        if actual_minimum_makespan = previous_minimum_makespan then
16           increase minimum_makespan_has_not_changed
17     crossover with probability Pc
18     mutation with probability Pm
```

3.2 Initialization and Evaluations

The population is formed by P_{size} individuals generated randomly. Many authors separate sequencing and assignment decisions in the HFS problems [20, 21]. We follow a way proposed by Ruiz and Maroto [11], where the assignment of jobs to machines at every stage is done by the evaluation function. In a HFS with no setup times and no availability constraints the first available machine would also result in the earliest completion time of the job. In a HFS with unrelated parallel machines we can find that the first available machine is very slow for a given job and thus assigning the job to this machine can result in a later completion time compared with assignment to other machines. With the addition of the setup times this problem can be even worse. To solve this problem, the jobs are assigned to the machine that can finish the job at the earliest time at a given stage, taking into consideration different processing speeds, setup times and machine availability.

3.3 Restart, Generational Scheme and Stopping Criterion

A restart mechanism based on the scheme proposed in [22] with modifications is applied.

- At each generation i, store the minimum makespan, mak_i,
- If $mak_i = mak_{i-1}$ then $countmak = countmak + 1$. Otherwise $countmak = 0$.
- If $countmak > G_r$ (number of generations without improve) then apply the following procedure:
  - Sort the population in ascending order of C_{max}.
  - Skip the 20% individuals from the sorted list (the best individuals)
  - From the remaining 80% individuals, 50% of them are replaced by simple SHIFT mutations of the best individual and 50% are replaced by newly randomly generated schedules.

- $regeneration = regeneration + 1$.
- $countmak = 0$.

The scheme is to replace some individuals in a new generation by individuals from the previous generation. As it was shown in [23], a steady state genetic algorithm where offspring replace the worst individuals in the population yielded much better results than regular genetic algorithms. The evaluations stop when the minimum makespan has not changed for 25 times ($G_r = 25$) and after $regeneration = 10$. Finally, the algorithm is executed 2 times.

3.4 Selection, Crossover and Mutation

For the parent's selection, the *tournament* selection, one of the classical selection schemes [24, 25], and the best for this kind of problems [11], is considered.

A mutation is incorporated to genetic algorithms to avoid convergence to local optimum, to reintroduce lost genetic material and variability in the population. Three mutation operators (insertion, swap and switch) widely used in the literature [11, 25, 26] are considered.

The crossover generates new sequences by combining two other sequences. The goal is to generate new solutions with better C_{max} values. Five operators from the literature are considered: OBX (Order Based Crossover) [26], PPX (Precedence Preservative Crossover) [27], OSX (One Segment Crossover) [5, 13], SB2OX (Similar Block 2-Point Order Crossover) [11] and TP (Two Point) [25]. In this paper, three new crossover operators are proposed: TPI, OBSTX, ST2PX.

TPI (Two Point Inverse). It is similar to Two Point crossover. It chooses two points randomly. Elements from the position 1 to the first point are copied from father 2. The positions from the first point to the second one are copied from the father 1. The positions from the second point to last one are copied from father 2.

OBSTX (Order based Setup Time Crossover). It comes from the original order based crossover taking in consideration sequence dependent setup times according to the binary mask. The value 1 of the mask indicates that the corresponding element of the father is copied to the son. The mask with value 0 indicates that the element of the father 2 is copied according the minimal sequence dependent setup time of machine choosen randomly at first stage.

ST2PX (Setup Time Two Point Crossover) works as follows. It chooses two crossover points randomly. Elements from the positions 1 to the first point are copied from father 1. Elements from the second point to last one are copied from father 1. Elements from the first point to second one are copied from the father 2 according to the minimal sequence dependent setup time of the machine chosen randomly at the first stage. Figure 1 shows how this operation is performed. Let us assume that the first point is the position 3, and the second point is the position 8 (Fig. 1A). The genes are copied from the position 1 to the position 3 of father 1 to the child. The genes from the position 8 to position 9 (last position) are copied from father 1 (Fig. 1B). The rest positions of the child are filled with best elements from the father 2, taking into account the sequence dependent

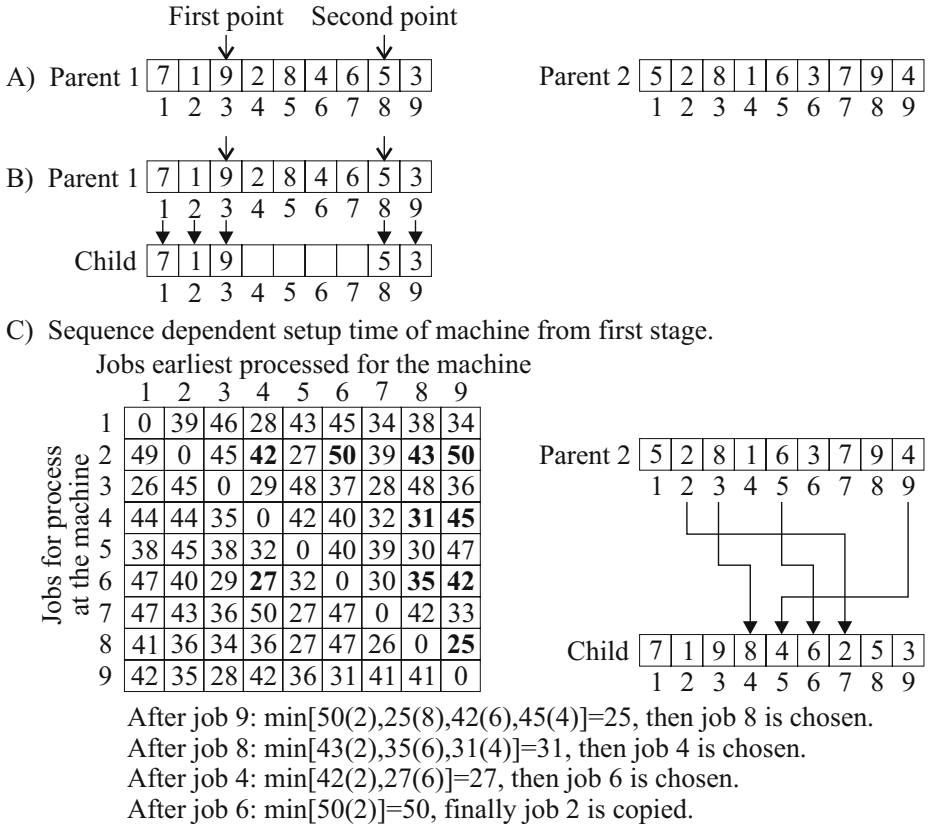

Fig. 1. Setup Time Two Point Crossover

setup times (Fig. 1C). For instance, to fill position 4 of the child (after the gene 9, position 3 of the child), four setup times (50, 45, 42, 25) are compared. The minimal value that corresponds to the gene 8 is copied to the position 4. The gene is equivalent to a job and represented as an element of each individual.

3.5 Experimental Setup

The following parameters are used to calibrate GA_{BC} algorithm: Population size (P_{size}): 50, 80, 100, 150 and 200; Crossover type: OBX, PPX, OSX, TP, SB2OX, TPI, OBSTX, ST2PX; Crossover probability (P_c): 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9; Mutation type: Insert, Swap, Switch; Mutation probability (P_m): 0.0, 0.005, 0.01, 0.015, 0.02, 0.03, 0.05, 0.1, 0.2, 0.4. Hence, 5.8.10.3.10 = 12000 different setups are considered. For each one, 360 problems, which are the part of full set of problems, as it was shown in [11], are solved. The number of stages (m) is set to 5, 10, and 20. The number of jobs is 20. For each

configuration, 10 different problems, where the processing times are uniformly distributed in the range 1 to 99, are considered. Three main groups of instances are defined where a given number of machines per stage is set. In the first one, there is a random uniformly distributed number of machines between one and three machines per stage. In the second group, there is a fixed number of two machines per stage. In the third one, three machines per stage is set. Inside each group there are four different subgroups of problems with different configurations for sequence dependent setup times. The setup times are drawn from the uniform distributions U[1, 9], U[1, 49], U[1, 99] and U[1, 124]. They correspond to 10%, 50%, 100% and 125% of the average processing times. Then, 12 different sets (three groups by four subgroups) with 30 problems of each one are used. The largest set up is the following: 20 jobs, 20 stages, with three machines per stage. Thus, 20x60 matrix of processing times, and 60 matrices of 20x20 of the setup times are used. The condition for termination is 25 iterations without improvement of the objective function. Then a regeneration procedure is executed 10 times, and the same parameter combination is applied 2 times taking the minimum makespan.

The advantage of the proposed algorithm is calculated by the following formula: $[(Heu_{sol} - Best_{sol})/Best_{sol}] \cdot 100$, where Heu_{sol} is the value of the objective function obtained by considered algorithm and $Best_{sol}$ is obtained by the best known one. Each instance for $Best_{sol}$ is evaluated 500,000 times with standard parameters proposed in [11]: ranking selection [24, 25], OP crossover (One point order crossover [25]), $P_c = 0.6$, $P_m = 0.01$, $P_{size} = 50$ and $G_r = 50$. Where, P_c is the cross probability, P_m is the mutation probability, P_{size} is the size of population, and G_r is the number of generations without improve.

4 Experimental Results

In this section, the computational results of the calibrated GA_{BC} algorithm with: ST2PX crossover, Swap Mutation, $P_c = 0.8$, $P_m = 0.4$, $P_{size} = 200$, binary tournament selection, and the stopping criterion described in Section 3.4 are presented. It is important to note that ST2PX is a new crossover for this problem.

As it was showed in Section 1, the unique algorithm GA_H proposed for this problem is introduced in [11], where 9 variants of metaheuristics [23, 28, 29, 30, 31,32,33,34] for the permutation flowshop problem are compared. GA_H is found to be the best for solving the problem. The GA_H is compared with GA_{BC} algorithm (Table 1). It can be seen that after a certain threshold of the problem complexity, the GA_H shows not satisfactory results with increasing the problem size. Only negative numbers in the table mean overcoming the best known result. The GA_{BC} algorithm shows superiority on 0.16% - 13.35% for the most problem cases.

Table 1. Average percentage of relative advantage over the best known solution

	SSD10_P13		SSD10_P2		SSD10_P3	
Instance	GA_{BC}	GA_H	GA_{BC}	GA_H	GA_{BC}	GA_H
20x5	0.18	0.06	-0.41	2.64	-1.25	3.06
20x10	-0.10	0.16	-0.72	1.76	-0.45	2.34
20x20	-0.55	0.22	0.09	1.23	-0.54	1.03
Average	-0.16	0.15	-0.35	1.88	-0.74	2.14
	SSD50_P13		SSD50_P2		SSD50_P3	
Instance	GA_{BC}	GA_H	GA_{BC}	GA_H	GA_{BC}	GA_H
20x5	-0.03	0.99	-5.38	3.10	-7.81	4.22
20x10	-2.40	0.16	-6.48	1.38	-6.05	2.86
20x20	-2.81	0.63	-4.58	1.13	-5.17	1.13
Average	-1.75	0.59	-5.48	1.87	-6.34	2.74
	SSD100_P13		SSD100_P2		SSD100_P3	
Instance	GA_{BC}	GA_H	GA_{BC}	GA_H	GA_{BC}	GA_H
20x5	-1.50	1.25	-9.77	4.09	-12.76	5.89
20x10	-5.92	1.13	-12.42	2.40	-11.86	2.20
20x20	-7.25	1.19	-10.13	1.64	-10.02	1.81
Average	-4.89	1.19	-10.77	2.71	-11.55	3.30
	SSD125_P13		SSD125_P2		SSD125_P3	
Instance	GA_{BC}	GA_H	GA_{BC}	GA_H	GA_{BC}	GA_H
20x5	-2.83	1.80	-11.44	3.96	-13.55	5.69
20x10	-8.05	1.56	-14.67	2.07	-14.35	3.67
20x20	-10.10	0.64	-12.66	1.46	-12.13	1.65
Average	-6.99	1.33	-12.92	2.50	-13.35	3.67
Average	-3.45	0.82	-7.38	2.24	-7.99	2.96

5 Conclusions

An effective genetic algorithm for a HFS with sequence dependent setup time, unrelated parallel machines at each stage, and machine availability constraints is presented. A crossover operator is introduced to improve the solution quality of a recently proposed genetic algorithm. Experimental results obtained on the benchmark data set show that the proposed algorithm can handle complex problems, yielding high quality solutions. Computational experiments are performed to compare the algorithm with other meta-heuristic approach: a previous best known genetic algorithm. The results are shown to be better than the previous ones. The algorithm achieved 6.27% better value of the objective function, in average, in all instances. Statistical tests applied to the results of these algorithms proved the superiority of the new variant of the genetic algorithm.

The results are not meant to be complete, but give an overview on the methodology and some interesting relations. These results motivate further explore the rationale for success of the algorithm as well as its robustness to larger problem sizes and especially in the real industry environment of the televisions production.

References

1. Morita, H., Shio, N.: Hybrid Branch and Bound Method with Genetic Algorithm for Flexible Flowshop Scheduling Problem. JSME International Journal, Series C 48(1), 46–52 (2005)
2. Kochhar, S., Morris, R.: Heuristic Method for Flexible Flow Line Scheduling. J. Manufacturing Systems 6(4), 299–314 (1987)
3. Santos, D., Hunsucker, J., Deal, D.: Global Lower Bounds for Flow Shops with Multiple Processors. European J. Operational Research 80, 112–120 (1995)
4. Aghezzaf, E., Artiba, A.: Aggregate Planning in Hybrid Flowshops. Int. J. Production Research 36(9), 2463–2477 (1998)
5. Guinet, A., Solomon, M.: Scheduling Hybrid Flowshops to Minimize Maximum Tardiness or Maximum Completion Time. Int. J. Production Research 34(6), 1643–1654 (1996)
6. Gupta, J., Tunc, E.: Minimizing Tardy Jobs in a Two-Stage Hybrid Flowshop. Int. J. Production Research 36(9), 2397–2417 (1998)
7. Portmann, M., Vignier, A.: Branch and Bound Crossed with GA to Solve Hybrid Flowshops. European J. Operational Research 107, 389–400 (1998)
8. Riane, F., Artiba, A., Elmaghraby, S.: A Hybrid Three-Stage Flowshop Problem: Efficient Heuristics to Minimize Makespan. European J. Operational Research 109, 321–329 (1998)
9. Arthanary, L., Ramaswamy, K.: An Extension of Two Machine Sequencing Problem. OPSEARCH. The Journal of the Operational Research Society of India 8(4), 10–22 (1971)
10. Salvador, M.: A solution to a special case of flow shop scheduling problems. In: Elmaghraby, S.E. (ed.) Symposium of the Theory of Scheduling and Applications, pp. 83–91. Springer, New York (1973)
11. Ruiz, R., Maroto, C.: A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European J. Operational Research 169, 781–800 (2006)
12. Vignier, A., Billaut, J., Proust, C.: Les Problmes D'Ordonnancement de Type Flow-Shop Hybride: tat de L'Art. RAIRO Recherche opérationnelle 33(2), 117–183 (1999)
13. Allaoui, H., Artiba, A.: Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. Computers & Industrial Engineering 47, 431–450 (2004)
14. Aghezzaf, E., Artiba, A., Moursli, O., Tahon, C.: Hybrid flowshop problems, a decomposition based heuristic approach. In: Proceedings of the International Conference on Industrial Engineering and Production Management, IEPM 1995, Marrakech. FUCAM – INRIA, pp. 43–56 (1995)
15. Zandieh, M., Fatemi Ghomi, S., Moattar Husseini, S.: An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. Applied Mathematics and Computation 180, 111–127 (2006)

16. Gourgand, M., Grangeon, N., Norre, S.: Metaheuristics for the deterministic hybrid flow shop problem. In: Proceedings of the International Conference on Industrial Engineering and Production Management, IEPM 1999, Glasgow. FUCAM - INRIA, pp. 136–145 (1999)
17. Adler, L., Fraiman, N., Kobacker, E., Pinedo, M., Plotnicoff, J., Wu, T.P.: BPSS: A Scheduling Support System for the Packaging Industry. Operations Research 41(4), 641–648 (1993)
18. Tang, L., Zhang, Y.: Heuristic Combined Artificial Neural Networks to Schedule Hybrid Flow Shop with Sequence Dependent Setup Times. In: Wang, J., Liao, X.-F., Yi, Z. (eds.) ISNN 2005. LNCS, vol. 3496, pp. 788–793. Springer, Heidelberg (2005)
19. Ryan, E., Azad, R.M.A., Ryan, C.: On the Performance of Genetic Operators and the Random Key Representation. In: Genetic Programming. LNCS, vol. 2003/2004, pp. 162–173. Springer, Heidelberg (2004)
20. Sherali, H., Sarin, S., Kodialam, M.: Models and algorithms for a two-stage production process. Production Planning and Control 1, 27–39 (1990)
21. Rajendran, C., Chaudhuri, D.: A multi-stage parallel processor flowshop problem with minimum flowtime. European J. Operational Research 57, 11–122 (1992a)
22. Alcaraz, J., Maroto, C., Ruiz, R.: Solving the multi-mode resource-contraints project scheduling problem with genetic algorithms. Journal of the Operational Research Society 54, 614–626 (2003)
23. Reeves, C.: A genetic algorithm for flowshop sequencing. Computers & Operations Research 22(1), 5–13 (1995)
24. Goldberg, D.: Genetic Algorithms in Search, optimization and Machine Learning. Addison-Wesley, Reading (1989)
25. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolutions Programs, 3rd edn. Springer, Heidelberg (1996)
26. Gen, M., Cheng, R.: Genetic algorithms & engineering optimization, p. 512. John Wiley & Sons, New York (1997)
27. Bierwirth, C., Mattfeld, D., Kopfer, H.: On permutation representations for scheduling problems. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 310–318. Springer, Heidelberg (1996)
28. Nawaz, M., Enscore Jr., E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA, The International Journal of Management Science 11(1), 91–95 (1983)
29. Osman, I., Potts, C.: Simulated annealing for permutation flow-shop scheduling. OMEGA, The Int. Journal of Management Science 17(6), 551–557 (1989)
30. Widmer, M., Hertz, A.: A new heuristic method for the flowshop sequencing problem. European J. Operational Research 41, 186–193 (1989)
31. Chen, C., Vempati, V., Aljaber, N.: An application of genetic algorithm for flow shop problems. European J. Operational Research 80, 389–396 (1995)
32. Murata, T., Ishibuchi, H., Tanaka, H.: Genetic algorithms for flowshop scheduling problems. Computers and Industrial Engineering 30(4), 1061–1071 (1996)
33. Aldowaisan, T., Allahvedi, A.: New heuristics for no-wait flowshops to minimize makespan. Computers & Operations Research 30, 1219–1231 (2003)
34. Rajendran, C., Ziegler, H.: Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. European J. Operational Research 155, 426–438 (2004)