International Journal of Industrial Engineering Computations 4 (2013) 345–354

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Optimizing a multi-objectives flow shop scheduling problem by a novel genetic algorithm

N. Shahsavari Pour[a], R. Tavakkoli-Moghaddam[b] and H. Asadi[c*]

[a]*Department of Industrial Management, Vali-e-Asr University, Rafsanjan, Iran*
[b]*Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran*
[c]*Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Kerman, Iran*

C H R O N I C L E

Article history:
Received December 10 2012
Received in revised format
March 15 2013
Accepted March 18 2013
Available online
March 25 2013

Keywords:
Flow-shop scheduling
Makespan
Total waiting time
Total tardiness
Multi-objective optimization
Genetic algorithm
ANOVA

A B S T R A C T

Flow-shop problems, as a typical manufacturing challenge, have become an interesting area of research. The primary concern is that the solution space is huge and, therefore, the set of feasible solutions cannot be enumerated one by one. In this paper, we present an efficient solution strategy based on a genetic algorithm (GA) to minimize the makespan, total waiting time and total tardiness in a flow shop consisting of n jobs and m machines. The primary objective is to minimize the job waiting time before performing the related operations. This is a major concern for some industries such as food and chemical for planning and production scheduling. In these industries, there is a probability of the decay and deterioration of the products prior to accomplishment of operations in workstation, due to the increase in the waiting time. We develop a model for a flowshop scheduling problem, which uses the planner-specified weights for handling a multi-objective optimization problem. These weights represent the priority of planning objectives given by managers. The results of the proposed GA and classic GA are analyzed by the analysis of variance (ANOVA) method and the results are discussed.

1. Introduction

The optimal solution to any flow-shop scheduling problem with n jobs and m machines determines the sequence of jobs on each machine with minimum makespan where each job is processed on different machines (1, 2, …, m) in an especial order. The number of possible schedules is $(n!)^m$ and the general problem is NP-hard (Fan & Winley, 2008). For this general problem, it is known that there is an optimal solution, in which the sequence of jobs is the same on the first two machines and the sequence of jobs is the same on the last two machines. Early research on flow shop problems was mainly based on the Johnson's theorem, which gives an appropriate procedure to locate an optimal solution with two or three machines with certain characteristics (Johnson, 1954; Kamburowski, 1997). The first proposed meta-heuristics for the permutation flow shop scheduling problem (PFSP) were the simulated annealing algorithms in Osman & Potts (1989) and Ogbu & Smith (1990). Espinouse et al. (1999) considered a two-machine no-wait flow shop scheduling problem by minimizing the makespan where the machine

* Corresponding author. Tel.: +989192421308
E-mail: hamed_assadi2000@yahoo.com (H. Asadi)

doi: 10.5267/j.ijiec.2013.03.008

availability was limited. Bertolissi (2000) presented a heuristic method for a no-wait flowshop scheduling problem, where the primary criterion was minimization of total flow time. Fink & Vob (2003) considered the application of different meta-heuristics including neighborhood search, simulated annealing and tabu search to solve a continuous flow-shop problem. They further examined the trade-offs between the running time and the solution quality as well as the knowledge and effort required to apply and calibrate their proposed method. Thornton and Hunsucker (2004) developed a new heuristic method, which minimizes the makespan in a flow shop scheduling problem with multiple processors and no intermediate storage. Bouquard et al. (2005) provided a complexity classification for different versions of a two-machine permutation flowshop scheduling problem for makespan minimization, in which some jobs were processed with no-wait in process.

Spieksma and Woeginger (2005) investigated a no-wait flow-shop paradox and demonstrated that increasing the speed of some machines in a no-wait flow-shop instance may increase makespan, significantly. Grabowski and Pempera (2005) investigated various local search techniques for a no-wait flowshop problem to minimize makespan. Kumar et al. (2006) proposed a Psycho-Clonal algorithm-based technique to solve a no-wait flowshop scheduling problem to minimize total flow times. Wang (2007) considered general, no-wait, and no-idle flowshop scheduling problems with deteriorating jobs. Oulamara (2007) presented a method for handling a task-scheduling problem in a no-wait flowshop with two batching machines by minimizing the makespan as primary objective. Su and Lee (2008) studied a two-machine flow-shop no-wait scheduling problem with a single server, in which the performance measure was the total completion time and presented both heuristic and branch-and bound (B/B) algorithms to tackle the given problem.

Additionally, some researchers have addressed multi-objective flowshop scheduling problems. Murata et al. (1996) considered a flowshop problem under two conditions including minimization of makespan and total tardiness penalties and minimization of makespan, total tardiness penalties and total flow times and used a multi-objective genetic algorithm to solve the resulted problems. Ponnambalam et al. (2004) proposed a multi-objective TSP-GA algorithm for a flow-shop scheduling problem where a weighted sum of multiple objectives, i.e. minimizing makespan, mean flow times and machine idle times, was considered. Toktas et al. (2004) considered a two-machine flowshop scheduling problem to minimize the makespan and maximize earliness. Ravindran et al. (2005) proposed three heuristic algorithms to solve a bi-objective flowshop scheduling problem where they minimized makespan and total flow times. Rahimi-Vahed and Mirghorbani (2007) devised a multi-objective particle swarm optimization for a bi-criteria flowshop scheduling problem to minimize the weighted mean completion time and weighted mean tardiness. They demonstrated that their method could outperform a multi-objective genetic algorithm or a set of representative problems, empirically. Tavakkoli-Moghaddam et al. (2007) proposed a multi-objective immune algorithm for a flowshop scheduling and they compared this algorithm with a conventional multi-objective genetic algorithm and introduced a new approach for solving the given problem. In this paper, planning managers were requested to prioritize the objectives of the scheduling then apply a new genetic algorithm (NGA) for solving the presented model.

The high speed of the algorithm and quick convergence of solutions make this approach suitable for large-scale scheduling problems with relatively large numbers of jobs. This paper is organized as follows. In Section 2, we present the problem definition and formulation. In Section 3, a solution procedure is introduced. We develop an algorithm, namely NGA to solve the problem. To examine the performance of the proposed NGA, a number of examples are presented and solved in Section 4. The related results of this algorithm are analyzed by the analysis of variance (ANOVA) method in Section 5. Finally, the remarking conclusion is given in Section 6.

2. Flow-shop scheduling problem

The flowshop scheduling problem consists of scheduling n jobs with given the processing times on m

machines and *n* jobs are processed on *m* machines with the same order, each job has to pass through each machine once and a job can be processed, , at the most, on one machine at a time. Each machine can perform only one interrupted job at a time. The objective of the problem is to detect an appropriate schedule to minimize the makespan and the total queue, simultaneously.

The primary objective is to minimize the job waiting time (i.e., waiting time length) before accomplishing the related operations. In some industries such as food and chemical, this goal plays essential role in planning and production scheduling, since there is a probability of the decay and deterioration of the products prior to accomplishment of operations in the workstation, when waiting time increases. In this paper, we present a new multi-objectives mathematical model for the flowshop scheduling problem. This scheduling problem is generally modeled based on the following assumptions,

- The processing times or operations on all machines are known and fixed.
- Setup times are included in the processing times and they are independent of the job position in the sequence of jobs.
- At a time, every job is processed on only one machine, and every machine processes only one job.
- The job operations on the machines may not be preempted.

We formulate the multi-objective flowshop scheduling problem using the following notations:

n	total number of jobs to be scheduled,
m	total number of machines in the process,
$t(_{i,j})$	processing time for job i on machine j (i=1,2,…,n) and (j=1,2,…,m)
P_i	job sequenced in the i-th position of a schedule,
C(P_i , j)	completion time of job P_i on machine j,
C(P_i , m)	completion time of job P_i on the m-th machine,
D(P_i , j)	waiting time for job i for machine j (i=1,2,…,n) and (j=1,2,…,m)
I(P_i , j)	tardiness of job P_i on machine j.

We can also calculate the completion times for an *n*-job, *m*-machine flowshop problem as follows,

$$C(P_1, 1)= t(1, 1) \quad (1)$$

$$C(P_1, j)=C(P_1 , j\ 1)+t(1, j); \quad j=1 \text{ to } m \quad (2)$$

$$C(P_i, j)=max\{C(P_{i\ 1}, j)\}+t(i, j); \quad i=1 \text{ to } n \ , \ j=1 \text{ to } m \quad (3)$$

$$Makespan= C(P_n, m) \quad (4)$$

$$C(P_i, j)= max\{0, C(P_{i\ 1}, j)\ \ C(P_1 , j\ 1) \quad (5)$$

$$\text{Total waiting time (TWT)}=\sum_{i=1}^{n}\sum_{j=2}^{m} D(P_i, j) \quad (6)$$

$$I(P_i, j)=\max\{0, C(P_1 , j\ 1)\ \ C(P_{i\ 1}, j) \quad (7)$$

$$\text{Total tardiness (TT)}=\sum_{j=1}^{n}\sum_{i=1}^{m} I(P_i, j) \quad (8)$$

In this paper, three objectives minimize the makespan (C(P_n, m)), the total waiting time (TWT)$=\sum_{i=1}^{n}\sum_{j=2}^{m} D(P_i, j)$ and the total tardiness ((TT)$=\sum_{j=1}^{n}\sum_{i=1}^{m} I(P_i, j)$), respectively.

$$\min z_1 = C(P_n, m), \quad (9)$$

$$\min z_2=\sum_{i=1}^{n}\sum_{j=2}^{m} D(P_i, j), \quad (10)$$

$$\min z_3 = \sum_{j=1}^{n} \sum_{i=1}^{m} I(P_i, j). \tag{11}$$

3. New genetic algorithm

The genetic algorithm (GA) was proposed by John Holland (1975). However, it has become one of the well-known meta-heuristics after Goldberg (1989). The mechanism of the simple GA is demonstrated in a pseudo code shown in Fig. 1.

(1) Randomly initialize a population of individual

(2) Perform a crossover operation to get offspring based on the probability of crossover

(3) Conduct a mutation based on the probability of mutation

(4) Fitness evaluation for each individual using an objective function

(5) Randomly select the survived chromosome for the next generation using roulette wheel

Fig. 1. Pseudo code of the simple genetic algorithm

There are various methods to improve the performance of the simple genetic algorithm. The first possibility is to implement the best configuration of the algorithm itself (Gen & Cheng, 1997; Pongcharoen et al., 2002). Alternatively, we could add in other heuristics as sub-process of the genetic algorithm, called hybrid GA (HGA). The most popular forms of the hybrid GA are to incorporate one or more of hill climbing and/or neighborhood search (Yamada & Reeves, 1998), optimization methods (e.g., neural network (Wang, 2005) or tabu search (Kido et al., 1993)), local search (Freisleben & Merz, 1996) or elitist strategy (Murata, 1996) as an add on extra to the simple GA loop of recombination and selection (Roach and Nagi, 1996). In this section, we present a new genetic algorithm (GA) to improve the performance of the simple genetic algorithm.

Because of the complexity of the flowshop problem, it is essential to develop an efficient evolutionary algorithm. In this paper, a new GA (NGA) is used to compute the optimal (or near-optimal) solution. The high performance of the algorithm and quick convergence of the solutions makes it possible to solve the above-mentioned problem. The proposed NGA is developed by using some modifications in the standard GA (Fig. 2).

3.1. NGA implementation

A chromosome is a set of integer values (genes), which represents the sequence of jobs. In the coding scheme, each job is represented by a gene and each set of *n* jobs is represented by a chromosome. The main steps of the NGA are as follows:

Step 1. The population size of each generation is set to be *N* and the proposed GA begins by randomly generating *N* chromosomes. This starting population is called generation 1.

Step 2. Determining the fitness function ($F(s)$) for each parent chromosome "*S*" by using the following equation.

$$F_{(S)} = w_c \times \frac{C_s - \mathrm{C}_{\min} + \gamma}{C_{\max} - \mathrm{C}_{\min} + \gamma} + w_D \times \frac{D_s - \mathrm{D}_{\min} + \gamma}{D_{\max} - \mathrm{D}_{\min} + \gamma} + w_I \times \frac{I_s - \mathrm{I}_{\max} + \gamma}{I_{\max} - \mathrm{I}_{\min} + \gamma}, \tag{12}$$

where w_c, w_D and w_I are the planner-specified weight and indicate the relative importance of the makespan, total waiting time and total tardiness, respectively. The values of weight coefficients (i.e., w_c, w_D and w_I) are subjectively selected in the range [0,1] by project managers, with $w_c + w_D + w_I = 1$.

C_{max}, C_{min}, D_{max}, D_{min} and I_{max}, I_{min} are the maximal and minimal values of makespan, total waiting time and total tardiness in the current population. γ is a very small positive number to prevent dividing by zero in the fitness function. It also does not permit the fitness function to be become zero because the model uses the inverse of the fitness function for the reproduction scale in the proposed NGA.

Step 3. To choose the parents in generation g in Step 1, we compute the normalized fitness based on Eq. (13) and Eq. (14), where m_g indicates the mean fitness of chromosomes in generation g. σ_g indicates the standard deviation of fitness in generation g and $z^g{}_i$ equals to the fitness normalized of the chromosome i. In the later and next step, due to the fact that the problem goals function is of the minimization type, chromosome (answer) of $z_i \leq 0$ are chosen as elite answer, and we add the answer having $z_i \geq 0$ to a list entitled the Black List.

Now, in order to escape from locating the algorithm in a local optimum, we should give a chance to the chromosomes in the black list to be selected. Thus, we use the crossover and mutation operators. To do so, we present the black list crossover rate and black list mutation rate. We employ the crossover operator for the black list chromosomes and add the children to the elite list. We use the mutation operator for each chromosome of the black list and add the new chromosomes to the elite list.

$$\sigma_g = \sqrt{\sum_{i=1}^{N} \frac{(fit_{(i)} - m_g)^2}{N-1}}, \tag{13}$$

$$z_i^g = \frac{fit(i) - m_g}{\sigma_g}. \tag{14}$$

Step 4. In this step, we generate offspring from parents (i.e., elite list) for entering the next generation. In this problem, for the number of jobs fewer than 50, a one-point crossover is implemented. The uniform crossover was introduced by Hansancebi and Erbatur (2000) and depending on the number of jobs, one point mutation is used. For instance, in a scheduling with nine jobs, two random chromosomes with feasible genes can be as follows:

Parent 1	2	1	5	3	6	4	7	9	8
Parent 2	4	3	2	5	4	1	3	2	5

For instance, since the number of jobs is small, one-point and two-point mutation are used. By applying the above-mentioned operators, the offspring produced from these parents are as follows:

One-cut-point crossover with random points (e_1=4).

Offspring 1	4	3	2	5	6	4	7	9	8
Offspring 2	2	1	5	3	6	1	7	9	8

Two point mutation with random points (e_1=6, e_2=9).

Offspring 1	2	1	5	3	6	8	7	9	4

It should be mentioned that the mutation rate ($Fm(g)$) decreases and uses Eq. (15), so that in the final generation, the mutation rate will be zero.

$$Fm_{(g)} = (Fm_{(g\text{-}1)}) \times \sigma_g \tag{15}$$

Step 5. Repeat Steps 2 to 4 until the σ_g=0 (i.e., the chromosomes do not change from one generation to the next one).

The steps of our proposed NGA are summarized in Fig. 2.

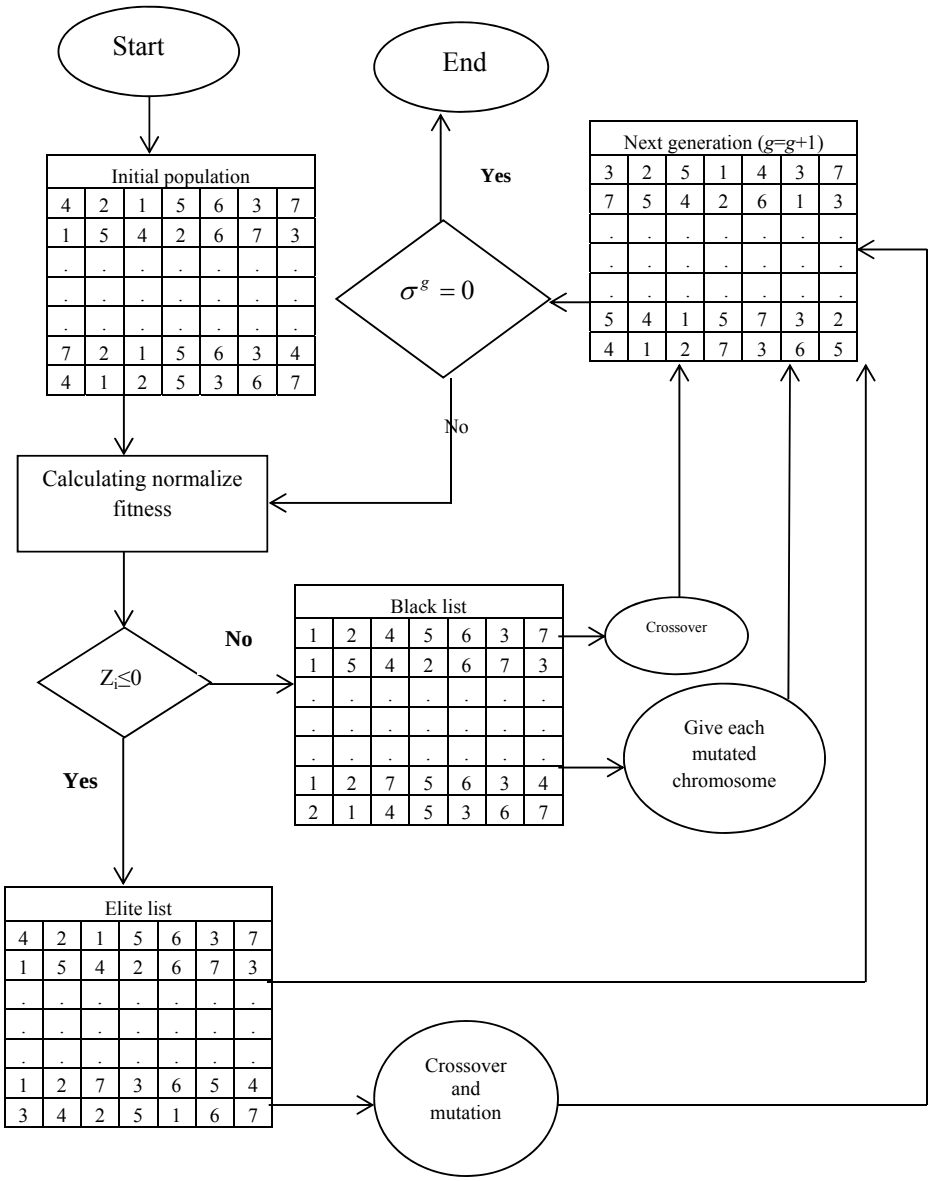

Fig. 2. NGA flowchart

4. Illustrative example

As an example, a two-machine flowshop scheduling problem with 10 jobs is presented in this section as depicted in Table 1. The model is programmed in the Microsoft Excel 2010 software using the Visual Basic Application (VBA). The project data given in Table 1 is entered in the application software. In this example, there are 3628800 solutions. The presented model is solved in order to obtain the optimal solution. The planner-specified weights are selected by the planning manager (W_c=0.2, W_D=0.4, W_I=0.4). Since the number of jobs is small in the given example, a combination of a one-point

crossover and black list crossover rate with pre-specified weights, two-point mutation and black list mutation rate have been used. The NGA parameters are set as follows.

G=100, *N*=70, two-point crossover rate=0.6, black list crossover rate=0.85, mutation rate=0.2, black list mutation rate=0.8.

The program is ran on a Pentium 4 PC with CPU 2.8 GHz, which chromosome [9,10,6,4,1,2,3,5,7,8] and its corresponding makespan, total waiting time and total tardiness (*C*=58, D=6, *I*=8) were obtained as the final solution. The planning manager may then obtain other optimum solutions by changing the value for the planner-specified weights, which are presented in the results in Table 2.

Table 1
Processing time for job *i* on machine *j*

Job	t(*i*, 1)	t(*i*, 1)
1	5	2
2	2	6
3	1	2
4	7	5
5	6	6
6	3	7
7	7	2
8	5	1
9	1	16
10	20	3

Table 2
Final outputs of the NGA

W_c	W_D	W_I	C	D	I	Solution Chromosome									
0.2	0.7	0.1	59	8	9	3	7	2	5	4	8	6	9	10	1
0.3	0.6	0.1	58	8	16	2	9	10	6	5	1	4	3	7	8
0.4	0.3	0.3	58	20	8	2	1	9	3	10	5	7	6	4	8
0.3	0.3	0.4	58	17	9	9	3	10	6	4	5	1	2	7	8
0.3	0.4	0.3	58	6	8	9	10	6	4	1	2	3	5	7	8
0.2	0.3	0.5	59	8	9	2	8	3	6	4	7	5	9	10	1
0.2	0.4	0.4	59	14	9	9	10	6	4	3	1	5	2	8	7
0.2	0.5	0.3	59	6	9	3	6	4	5	9	10	8	1	2	7
0.1	0.6	0.3	60	11	10	9	10	4	2	5	8	1	6	7	3

5. Experimental evaluation

This section evaluates the performance of our proposed NGA and the classical GA. These algorithms are coded and implemented in Excel 2010 by the VBA and are ran on a Pentium 4 PC with CPU 2.8 GHz and 512 MB of RAM memory. We use the weight relative deviation (WRD) as a common performance measure to compare these algorithms that is computed by:

$$WRD = w_C \times \frac{C_{alg} - C_{min}}{C_{min}} + w_D \times \frac{D_{alg} - D_{min}}{D_{min}} + w_I \times \frac{I_{alg} - I_{min}}{I_{min}}, \tag{16}$$

where C_{alg} , D_{alg} and I_{alg} are the makespan, total waiting time and total tardiness for a given algorithm, respectively. C_{min} , T_{min} and Q_{max} are the best solutions obtained by each algorithm for a given instance, respectively. The NGA and GA are implemented with same parameters for twenty five times. Their

results are analyzed via the analysis of variance (ANOVA) method. The means plot and least significant different (LSD) interval for the NGA and GA are shown as Fig. 3. It demonstrates that the NGA gives better outputs than the GA for the given problem statistically.

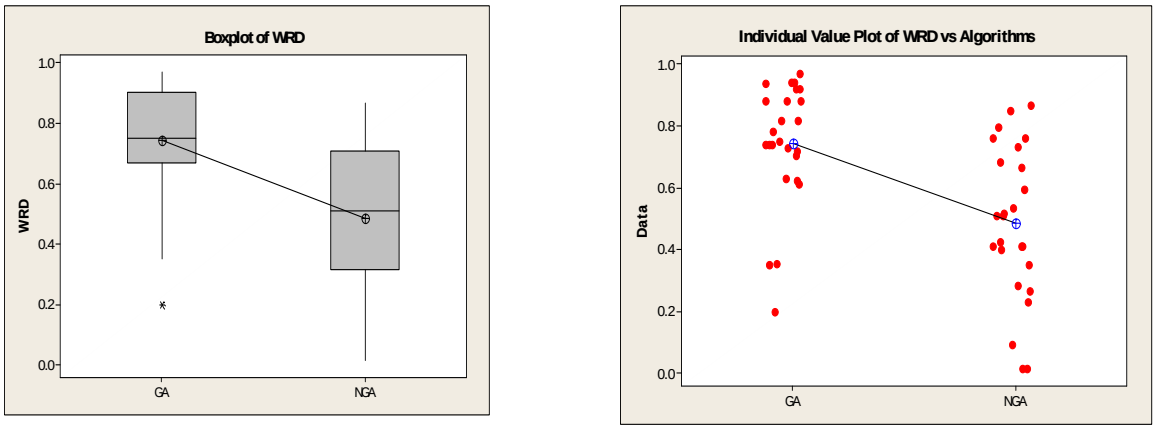

Fig. 3. Means plot and LSD intervals for the NGA and GA

6. Conclusion

In this paper, we have presented a multi-objective technique to consider flowshop problem where the objectives include makespan, total waiting time and total tardiness in term of the planner-specified weights. These weights have represented the priority of scheduling objectives, which must be selected by the managers and by choosing different weights we have managed to find various efficient solutions. To solve the given problem, a novel genetic algorithm (NGA) has been also developed. The planning managers have been asked to prioritize the objectives of the scheduling, and then we have applied the proposed new genetic algorithm (NGA) for solving the presented model. The high speed of the proposed algorithm and its quick convergence makes it desirable for large scheduling with a large number of jobs. Furthermore, we have used the weight relative deviation (WRD) measure to compare the performance of the NGA and GA by the ANOVA method. By considering uncertainty in processing time, this model can be extended to the cases which can be more realistic.

References

Bertolissi, E. (2000). Heuristic algorithm for scheduling in the no-wait flow-shop. *Journal of Materials Processing Technology*, (107) 459–465.

Bouquard, J.L., Billaut, J.C., Kubzin, M.A., & Strusevich, V.A. (2005). Two-machine flow-shop scheduling problems with no-wait jobs. *Operations Research Letters,* (33) 255–262.

Espinouse, M.L., Formanowicz, P., & Penz, B. (1999). Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability. *Computers and Industrial Engineering*, (37) 497–500.

Fink, A., & Vob, S. (2003). Solving the continuous flow-shop scheduling problem by meta-heuristics. *European Journal of Operational Research*, (151) 400–414.

Freisleben, B., & Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric travelling salesman problems. *Proceedings of IEEE International Conference on Evolutionary Computation*, 159–164.

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. John Wiley & Sons, New York.

Grabowski, J., & Pempera, J. (2005). Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Computers and Operations Research*, (32), 2197–2212.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-

Wesley, Massachusetts.

Hansancebi, O., & Erbatur, F. (2000). Evaluation of crossover techniques in genetic algorithm based optimum structural design. *Computer and Structures*, (78), 435–448.

Holland, J. (1975). Adaptation in natural and artificial systems. *University of Michigan Press: Ann Habor*.

Johnson, S.M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, (1) 61–68.

Kamburowski, J. (1997). The nature of simplicity of Johnson's algorithm. *Omega – International Journal of Management Science*, (25), 581–584.

Kido, T., Kitano, H., & Nakanishi, M. (1993). A Hybrid Search for Genetic Algorithms: Combining Genetic Algorithms, TABU Search, and Simulated Annealing. *Proceedings of the 5th International Conference on Genetic Algorithms,* San Mateo, CA, 641.

Kumar, A., Prakash, A., Shankar, R., & Tiwari, M.K. (2006). Psycho-Clonal algorithm based approach to solve continuous flow-shop scheduling problem. *Expert Systems with Applications*, (31), 504–514.

Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Multi-objective genetic algorithm and its applications to flow-shop scheduling. *Computers and Industrial Engineering*, (30), 957–968.

Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Genetic algorithms for flow-shop scheduling problems. *Computer & Industrial Engineering*, (30), 1061-1071.

Ogbu F.A., & Smith D.K. (1990). The application of the simulated annealing algorithms to the solution of the n / m / C_{max} flow-shop problem. *Computers & Operations Research,* (17), 243–253.

Osman, I.H., & Potts, C.N. (1989). Simulated annealing for permutation flow-shop scheduling. OMEGA, *International Journal of Management Science*, (17), 551–557.

Oulamara, A. (2007). Makespan minimization in a no-wait flow-shop problem with two batching machines. *Computers and Operations Research*, (34), 1033–1050.

Ponnambalam, S.G., Jagannathan, H., Kataria, M., & Gadicherla, A. (2004). A TSP-GA multi-objective algorithm for flow-shop scheduling. *International Journal of Advanced Manufacturing Technology*, (23), 909–915.

Pongcharoen, P., Hicks, C., Braiden, P.M., & Stewardson, D.J. (2002). Determining optimum genetic algorithm parameters for scheduling the manufacturing and assembly of complex products. *International Journal of Production Economics*, (78), 311–322.

Rahimi-Vahed, A.R., & Mirghorbani, S.M. (2007). A multi-objective particle swarm for a flow-shop scheduling problem. *Journal of Combinatorial Optimization,* (13), 79–102.

Ravindran, D., Noorul Haq, A., Selvakuar, S.J., & Sivaraman, R. (2005). Flow-shop scheduling with multiple objective of minimizing makespan and total flow time. *International Journal of Advance Manufacturing Technology*, (25), 1007–1012.

Roach, A., & Nagi, R. (1996). A hybrid GA-SA algorithm for just-in-time scheduling of multi-level assemblies. *Computers & Industrial Engineering*, (30), 1047–1060.

Spieksma, F.C.R., & Woeginger, G.J. (2005). The no-wait flow-shop paradox. *Operation Research Letter*, (33), 603–608.

Su, L.-H., & Lee, Y.-Y. (2008). The two-machine flow-shop no-wait scheduling problem with a single server to minimize the total completion time. *Computers and Operations Research*, (35), 2952-2963.

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A.R., & Mirzaei, A.H. (2007). Solving a bi-criteria permutation flow-shop problem using immune algorithm. *Proceedings of the First IEEE Symposium on Computational Intelligence*, vol. 1, Honolulu, Hawaii, (April 2007), pp. 4

Thornton, H.W., & Hunsucker, J.L. (2004). A new heuristic for minimal makespan in flow-shops with multiple processors and no intermediate storage. *European Journal of Operational Research*, (152), 96–114.

Toktas, B., Azizoglu, M., & Koksalan, S.K. (2004). Two-machine flow-shop scheduling with two criteria: Maximum earliness and makespan. *European Journal of Operational Research,* (157), 286–295.

Wang, J.B. (2007). Flow-shop scheduling problems with decreasing linear deterioration under

dominant machines. *Computers and Operations Research*, (34), 2043–2058.

Wang, L. (2005). A hybrid genetic algorithm-neural network strategy for simulation optimization. *Applied Mathematics and Computation*, (170), 1329-1343.

Yamada, T., & Reeves, C.R. (1998). Solving the sum permutation flow-shop scheduling problem by genetic local search. *Proceedings of IEEE International Conference on Evolutionary Computation*, 230- 234.