International Journal of Applied Operational Research
Vol. 5, No. 2, pp. 49-60, Spring 2015

Journal homepage: ijorlu.liau.ac.ir

A Mathematical Programming Model and Genetic Algorithm for a Multi-Product Single Machine Scheduling Problem with Rework Processes

R. Ramezanian*

Received: 22 September 2014 ; **Accepted:** 12 February 2015

Abstract In this paper, a multi-product single machine scheduling problem with the possibility of producing defected jobs, is considered. We concern rework in the scheduling environment and propose a mixed-integer programming (MIP) model for the problem. Based on the philosophy of just-in-time production, minimization of the sum of earliness and tardiness costs is taken into account as the objective function. It is possible to obtain optimal solutions for small-sized problems using the MIP model by operation research solvers. Due to the complexity of the problem, exact algorithms are inefficient for medium and large-sized problems. For large-sized problems, an adapted genetic algorithm (GA) is used to solve them. The implemented GA is compared to the optimal solutions generated by an optimization solver, and to the solutions generated by dispatching rules procedure. Computational experiments are performed to illustrate the efficiency of the adapted GA algorithm.

Keywords: Single machine scheduling, Rework process, Mathematical programming, Genetic algorithm.

1 Introduction

Scheduling problem is one of the important issues affecting the productivity of production systems. This problem deals with the determination of optimum sequence of manufacturing jobs with respect to different sequencing patterns. In traditional scheduling problems, it is assumed that the jobs are manufactured without any defects in the production stages. But, in many real-world production environments, it is reasonable to assume some of the items can be defective due to non-perfect technology, damaged machines, an unsteady production environment, an unpredictable condition or human mistakes. Thus, an entire production process includes two stages that we call work and rework. Rework activities are defined as all of the activities required to transform items that have not been manufactured according to predefined standard qualifications [1]. In order to improve and regain defective jobs, they are putted into recovery processes. In this paper, it is assumed that the number of recovery processes is finite.

* **Corresponding Author.** (✉)
E-mail: Ramezanian@kntu.ac.ir (R. Ramezanian)

R. Ramezanian
Department of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran.

Rework is an important issue in many process industries such as semiconductor, glass and steel manufacturing [1]. Flapper et al. [1] focused on planning and control of rework activities in the process industries and presented a comprehensive survey of the related issue. Flapper and Jensen [2] and Inderfurth and Teunter [3] integrated rework and production processes to challenge planning and control problems, especially if both processes are using the same equipment.

A lot of researches have investigated the effect of rework on manufacturing processes. Most studies in this area are relevant to determine the lot size to obtain the economic batch quantity for products in the production environments [4-7]. Teunter and Flapper [4] handled the problem which determines the optimal batch sizes to maximize an average profit in a manufacturing system with single stage single product. In this production system rework process starts when a fixed number of rework products is gathered. Inderfurth et al. [6, 7] considered scheduling of work and rework processes on a single facility in which the same products are produced in batches. The problem is to find batch sizes such that the demands for the items are satisfied and total cost is minimized. They proposed polynomial time algorithms based on dynamic programming to solve this problem.

Recently, researches on rework policies that are combined with dispatching rules have increased in number. Kuhl and Laubisch [8] determined a dispatching rule and a rework strategy as significant factors which affect productivity in semiconductor manufacturing. They looked for the best combination among five dispatching rules and three rework strategies using simulation. Sha et al. [9] studied the finding of a dispatching rule for integrating the rework strategies in the photolithography area of wafer fabrication. They proposed a dispatching rule (Rw-DR) that determines both original lots and rework lots. Mean flow time, on-time delivery, mean tardiness, work in process, and mean flow time of rework lots were used as different performance indicators for the results of proposed approach. Shin and kang [10] focused on the problem of scheduling jobs on parallel machines considering rework probabilities, due-dates and setup times. They proposed a greedy rework probability with due-dates (GRPD) algorithm focusing on the rework processes to solve the problem. The performance of proposed dispatching rule is measured by six indicators. Kan and shin [11] considered parallel machines scheduling problem with rework probabilities, due-dates, and setup times. They developed two heuristic approaches based on a dispatching algorithm named minimum rework probabilities with due-dates (MRPD) and problem-space-based search (PSBS) method. In order to evaluate the effectiveness of the algorithms, six performance indicators including total tardiness, maximum lateness, mean flow-time, mean lateness, the number of tardy jobs, and the number of reworks were used.

Some researchers considered the case in which reworkable jobs have to undergo some deterioration processes while they wait to be recovered. Deterioration results in an incensement in processing time as well as in processing cost for reworking the defective items. Thus, planning and control activities can be complicated considerably [1, 12-14]. Tayebi Araghi et al. [15] studied flexible job-shop scheduling with sequence-dependent set-ups, learning effect and deterioration in jobs. They implemented hybrid meta-heuristic algorithm based on genetic algorithm and the variable neighborhood search to solve the problem. Yin et al. [16] investigated a real-life scheduling situations in which the jobs deteriorate at a certain rate while waiting to be processed. They considered the actual time for processing a job depends not only on the starting time of the job but also on its scheduled position. Meta-heuristic algorithms including have been proposed for solving the machine scheduling problem [15,17-20]. Ramezanian and Saidi-Mehrabad [21] studied a multi-product unrelated parallel machines scheduling problem with the possibility of producing imperfect

jobs. They presented a MIP model to formulate the problem and developed some heuristics focusing on the rework processes to solve the problem.

The scheduling environments dealt with in this paper can be briefly summarized as follows. There are a number of jobs to be processed on single machine considering defective items, rework probabilities, due-dates, earliness and tardiness costs. When a job finishes, it is tested as to whether the job will satisfy the preset qualification or not. If a job failed at the test, the job should reenter the work center for rework process until it passes the test. In this paper, it is assumed that the number of recovery processes is finite. Also, it is assumed that rework probability for each job on the machine can be estimated through historical data acquisition. The problem configuration is illustrated in Fig. 1.

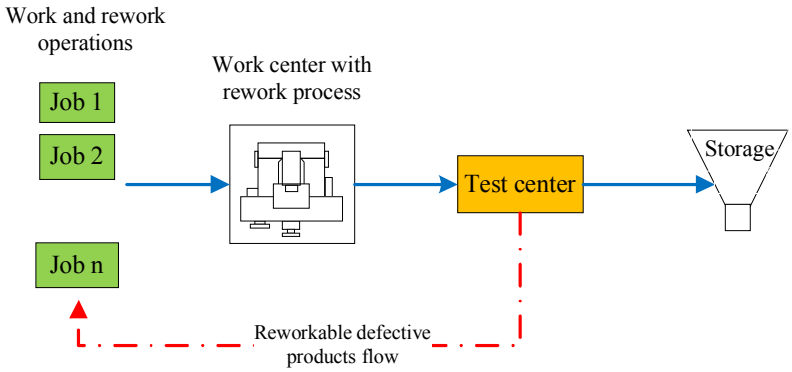

Fig. 1 Single machine scheduling with rework processes configuration

In this paper, a multi-product single machine scheduling problem is considered. We assume that defective items can be produced in any production run. We propose a mixed-integer programming model for this problem.

The rest of the paper is organized as follows: Section 2 describes mixed integer programming formulation of the problem. In the Section 3, we present implemented genetic algorithm to solve our proposed model. Section 4 presents the acquired computational results and, finally, Section 5 provides conclusions and future research suggestions.

2 Mathematical formulation

In this section, a mathematical formulation model to schedule work and rework operations on single machine scheduling environment is presented. The assumptions, parameters and decision variables used throughout the paper are detailed as the following.

General assumptions:

- There is only one machine.
- A job has some operations which the first one is main process and the others are rework processes that are probabilistic.
- Setup times between jobs are negligible or included in the processing times (sequence-independent setup times).
- Machine is available at all times.
- All jobs are available for processing at time zero.
- Machine cannot process two jobs at the same time.

- Each job is processed on at most one machine at a time.
- No preemption of jobs is allowed (once the processing of a job on a machine has started, it cannot be interrupted on that machine).
- There is no precedence constraint among the jobs.
- No more than one operation of the same job can be executed at a time.
- Any breakdowns and scheduled maintenance are not allowed.
- The processing times are independent of the sequence and are given.

Parameters:

n: Number of jobs
i: Job index, $i = 1,...,n$
j: Process index, $j=1,..., l_i$ ($j=1$: main process; $j=2,...,l_i$: rework process) $K = \sum_{i=1}^{n} l_i$
p_{ij}: Defective probability of jth operation for job i ($p_{il_i} = 0$). It is assume that after (l_i-1) recovery process (rework) the piece will have good quality.
t_i: Processing time of main process for job i
α_j: Decreasing coefficient of processing time in rework process j ($0 \leq \alpha_j \leq 1$)
t_{ij}: Processing time of jth operation for job i $t_{i1}=t_i$, $t_{ij}=t_{i(j-1)}(1-\alpha_j)$
E_i: Earliness of job i $E_i = max\{d_i - q_i, 0\}$
T_i: Tardiness of job i $T_i = max\{q_i - d_i, 0\}$
d_i: Due date of job i
H_i: Holding cost of job i per time unit
β_i: Shortage cost of job i per time unit

Decision variables:

q_i: Completion time of job i
C_{ij}: Completion time of jth operation for job i
x_{ijk}: Binary variable taking value 1 if jth operation for job i on machine is processed in kth order and 0 otherwise.

Objective function:

Due to the philosophy of just-in-time production, both earliness and tardiness penalties are used in objective functions in the model. Considering both of tardiness and earliness penalties is in order to satisfy the requests of real world production environments. The earliness cost can represent the inventory cost for early finished stocks, and the tardiness cost can represent the penalty cost for the late delivery.

To determine the expected value of completion time for jth operation of job i (C_{ij}), the equation (1) is used:

$$E(C_{ij}) = \sum_{k=2}^{K} \left(\sum_{i'=1}^{n} \sum_{k'=1}^{k-1} (t_{i'1} x_{i'1k'}) + \sum_{i'=1}^{n} \sum_{j=2}^{l_i} \sum_{k'=1}^{k-1} (t_{i'j} x_{i'jk'} \prod_{j'=1}^{j-1} p_{ij'}) \right) x_{ijk} + t_{ij} \; ; \quad \forall i, j \tag{1}$$

The expected value of completion time for job i can be computed based on Bayes' theorem, as equation (2):

$$E(q_i) = E(C_{i1} \,|\, Completed\ at\ O_{i1})(1 - p_{i1}) + E(C_{i2} \,|\, Completed\ at\ O_{i2})[p_{i1}(1 - p_{i2})] + ... +$$
$$E(C_{il_i} \,|\, Completed\ at\ O_{il_i})(1 - p_{il_i})(\prod_{j=1}^{l_i - 1} p_{ij}) = C_{i1}(1 - p_{i1}) + \sum_{j=2}^{l_i} C_{ij}(1 - p_{ij})(\prod_{j'=1}^{j-1} p_{ij'}) \tag{2}$$

Eventually, our proposed MIP model can be stated as follow:

$$Min \quad z = \sum_{i=1}^{n} (H_i E_i + \beta_i T_i) \tag{3}$$

s.t.

$$q_i + E_i - T_i = d_i; \quad \forall i \tag{4}$$

$$C_{ij} = \sum_{k=2}^{K} (\sum_{i'=1}^{n} \sum_{k'=1}^{k-1} (t_{i'1} x_{i'1k'}) + \sum_{i'=1}^{n} \sum_{j=2}^{l_i} \sum_{k'=1}^{k-1} (t_{i'j} x_{i'jk'} \prod_{j'=1}^{j-1} p_{i'j'})) x_{ijk} + t_{ij}; \quad \forall i, j \tag{5}$$

$$q_i = C_{i1}(1 - p_{i1}) + \sum_{j=2}^{l_i} C_{ij}(1 - p_{ij})(\prod_{j'=1}^{j-1} p_{ij'}); \quad \forall i \tag{6}$$

$$\sum_{k=1}^{K} x_{ijk} = 1; \ \forall i, j \tag{7}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ijk} = 1; \ \forall k \tag{8}$$

$$\sum_{k=1}^{K} x_{ijk} \le \sum_{k=1}^{K} x_{i(j-1)k}; \ \forall i, j = 2,...,l_i \tag{9}$$

$$\sum_{k=1}^{l} x_{i(j-1)k} \le \sum_{k=l}^{K} x_{ijk}; \ \forall i; j = 2,...,l_i; l = 2,...,K \tag{10}$$

$$q_i \ge 0; \ \forall i \tag{11}$$

$$C_{ij} \ge 0, \forall i, j \tag{12}$$

$$x_{ijk} = \{0,1\}; \ \forall i, j, k \tag{13}$$

The objective function (3) minimizes total earliness and tardiness costs. The constraint set (4) determines earliness and tardiness of each job. The constraint set (5) determines the expected value of completing time of each operation for each job. The constraint set (6) determines the expected value of completing time for each job. The constraint set (7) insures that each operation for each job is assigned to only one position (priority) in the operation sequence and the constraint set (8) certifies that each sequence position is filled with only one operation of a job. The constraint set (9) insures that each operation of a job can be started after previous operation of the same job is completed. The constraint set (10) guarantees that position (priority) of each operation of a job is greater than the position (priority) of previous operation of the same job. (11) to (13) are logical constraints.

3 The genetic algorithm

Genetic algorithms (GA) are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, crossover, mutation, and selection. Gas are used to obtain good-quality solution for constrained optimization and combinatorial optimization problems. Following is the presented genetic algorithm.

3.1. Schematic structure of genes

In this paper, each gene is an operation and the chromosome is operation sequence vector (Solution). For scheduling problem, Gen et al. [22] proposed an operation sequences representation: they name all operations for a job with the same symbol and then interpret them according to the order of occurrence in the sequence of a given chromosome. The *i*th job appears in the operation sequence vector (v) exactly l_i times to represent its l_i ordered operations.

Imagine a single machine shop scheduling with rework problem with three jobs, where each job requires four operations. The first operation for each job is main process and other operations are rework processes that are probabilistic. The operation sequence for this example which is represented in Fig. 2 can be translated into a list of ordered operations below:

$$O_{21} \succ O_{11} \succ O_{22} \succ O_{23} \succ O_{31} \succ O_{12} \succ O_{32} \succ O_{33} \succ O_{13} \succ O_{24} \succ O_{14} \succ O_{34}$$

Priority (k)	1	2	3	4	5	6	7	8	9	10	11	12
Operation Sequence: v(k)	2	1	2	2	3	1	3	3	1	2	1	3

Fig. 2 Illustration of the operation sequence vector

The main advantage of the vector Gen et al.'s [22] representation is that each possible chromosome always represents a feasible solution candidate.

3.2 Initialization

To generate the initial population, random sequence generation is used. Thus, the diversity of the initial population can be retained.

3.3 Selection

During each successive generation, a proportion of the existing population is selected to breed offspring. The selection procedure provides the opportunity to deliver the genes of a good solution to next generation. In the literature, there are various selection operators available that can be used to select the parents. In this study, the tournament selection is used.

3.4 The genetic operators

3.4.1 Crossover

Crossover operator recombines two chromosomes to generate a number of children. Offspring of crossover should represent solutions that combine substructures of their parental solutions. Compared to other meta-heuristic methods, such as: tabu search (TS), simulated annealing (SA), ant colony optimization (ACO), crossover may be the most distinct operation of GAs, making heritability especially critical [23].

The enhanced order crossover expanded from the classical order crossover [24] works as follows:

Step1. Randomly choose two chromosomes, named parent 1 and parent 2:

Step 1.1. Two chromosomes are selected randomly, then the chromosome with lower fitness value is chosen and named parent 1.

Step 1.2. Repeat Step 1.1, then name the chromosome with lower fitness function parent 2.

Step 2. Randomly select a subsection of operation sequence from parent 1.

Step 3. Produce a proto-child by copying the substring of operation sequence into the corresponding positions.

Step 4. Starting with the first position of parent 2, delete the operations which are in the substring from the second parent. The resulted sequence of operations contains the operations that the proto-child needs.

Step 5. Put the remaining operations into the empty positions of the proto-child from left to right according to the order of the sequence in the second parent.

The implemented crossover procedure is illustrated in Fig. 3.

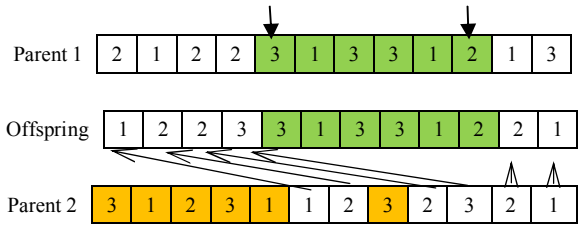

Fig. 3 Illustration of the enhance order crossover

3.4.2 Mutation

Swap mutation is used to mutate the individuals and its mechanism works as follows:

Step 1. Randomly choose one chromosome.

Step 2. Randomly choose two priorities from selected chromosome in step 1.

Step 3. Replace selected operations with each other.

The used mutation procedure is illustrated in Fig. 4.

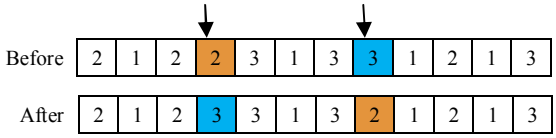

Fig. 4 Illustration of swap mutation

3.5 Fitness Function

The fitness function is the same as the objective function which defined in Section 2. In the proposed genetic algorithm the lower fitness function is desired.

3.6 Stopping condition

The GA continues to process the above procedure until the termination condition is met. The search process terminates if the number of iterations is greater than maximum number of generations, a predetermined constant set by user.

4 Computational results

In this section, we provide an experimental design to compare the results of exact, heuristic and meta-heuristic algorithms. A set of experiments are conducted for cases in which the number of jobs ranges from 2 to 100 and the number of operations for the single machine ranges from 2 to 5. For each configuration, the algorithm was run with randomly assigned processing times, due dates, inventory costs, shortage costs and rework probabilities. Each data set was run 5 times and the best trial is taken as the objective value obtained.

For each problem, the job processing times, due dates, inventory costs and shortage costs are uniform [10, 100], [500, 2500], [5, 25] and [5, 25], respectively. Defective probability for the processing of jobs on the machine is a random real number generated from [0.2, 0.4]. Decreasing rate of recovery processing time is set to be 0.3.

The GA and heuristic methods are coded in MATLAB R2007(b) and all tests are solved on a PC with an Intel Core Duo (2GHz) CPU and 1GB RAM. For each run, the number of generations is set as 100, the population number, crossover rate and mutation rate in each generation are 100, 0.9 and 0.1.

For small-sized problems, the performances of GA are compared with optimal solutions and the solutions generated by random sequence (Random), modified shortest processing time (M-SPT) and modified early due date (M-EDD) dispatching rules. Table 1 shows detailed numbers of average objective function (OF) and CPU time. For problems 1 and 2 from Table 1, the optimal value is obtained by Lingo optimization solver and GA. For problems 3-7, the objective function is obtained by Lingo after one hour is recorded. For these problems, the optimal solution could not be found within an hour. As shown in Table 1, the objective value obtained by the implemented GA is lower than the results of Lingo and dispatching rules. Table 2 shows the paired T-test for comparison of exact solution obtained by Lingo and GA. This table shows that there is no statistically difference between exact solutions and GA results when confidence level is set at 0.95 (with p-value = 0.08).

Table 1 Comparison of all small-sized problems

Prob.	n	*l*	Lingo		Random		M-SPT		M-EDD		Adapted GA	
			FF	Time(s)	FF	Time(s)	FF	Time(s)	FF	Time(s)	FF	Time(s)
1	2	2	**679.14**	<1	1712.12	0.011	1712.12	0.011	1700.93	0.011	**679.14**	0.178
2	3	2	**2135.88**	13	3986.59	0.012	4113.32	0.012	3986.59	0.012	**2135.88**	0.261
3	3	3	570.51	1 hour	4164.23	0.013	2946.81	0.012	3100.18	0.013	**498.90**	0.298
4	4	2	2052.40	1 hour	3982.05	0.012	5470.06	0.012	5415.26	0.012	**1576.83**	0.294
5	4	3	1599.85	1 hour	3818.85	0.012	3364.71	0.012	3364.71	0.013	**1572.28**	0.416
6	5	2	909.58	1 hour	5614.56	0.013	7860.24	0.012	7484.26	0.013	**623.83**	0.362
7	5	3	3856.67	1 hour	10834.86	0.013	10491.42	0.012	13967.92	0.013	**3727.21**	0.524

Table 2 Paired T-Test for comparison of exact solution and GA

	N	Mean	StDev	SE Mean
Exact solution	7	1686.29	1148.68	434.16
GA	7	1544.87	1140.58	431.1
Difference	7	141.423	178.169	67.342
95% CI for mean difference: (-23.356, 306.202)				
T-Value = 2.10 P-Value = 0.08				

Due to the complexity of the problem, exact algorithms are inefficient for medium and large-sized problems. For these problems, the performances of GA are compared with random sequence, M-SPT and M-EDD procedures. The experimental results for the problems are listed in Table 3 and showed in Fig. 5. Since, the results show that the performance of the presented GA is satisfactory and can reach good-quality solutions within a reasonable computational time (See Table 3 and Fig. 5).

Table 3 Comparison of all large-sized problems

Prob.	n	*l*	Random		M-SPT		M-EDD		GA	
			FF	Time (s)	FF	Time (s)	FF	Time (s)	FF	Time (s)
1	20	3	389,095.57	0.027	434,564.31	0.026	429,447.16	0.026	**282,645.88**	4.85
2	20	5	334,681.16	0.027	345,414.43	0.027	224,060.25	0.028	**187,545.28**	10.63
3	30	3	383,449.49	0.027	345,186.71	0.027	519,619.98	0.028	**42,401.11**	10.50
4	30	5	490,481.51	0.030	606,180.92	0.029	383,646.07	0.030	**251,128.43**	23.15
5	40	3	612,220.12	0.028	526,408.41	0.029	785,264.97	0.029	**82,334.62**	18.96
6	40	5	450,104.82	0.033	826,284.55	0.032	123,813.22	0.033	**119,402.73**	40.48
7	50	3	882,406.39	0.109	915,375.41	0.030	1,191,184.53	0.030	**136,326.89**	28.05
8	50	5	541,266.22	0.037	664,876.51	0.036	288,747.52	0.036	**270,453.28**	60.98
9	60	3	1,073,814.56	0.033	1,190,987.40	0.031	1,625,574.25	0.032	**466,723.52**	39.66
10	60	5	813,789.82	0.041	968,325.57	0.040	475,031.64	0.040	**346,781.07**	88.57
11	70	3	1,512,502.49	0.035	1,736,351.75	0.035	2,155,182.95	0.034	**712,166.95**	52.92
12	70	5	1,310,186.17	0.046	1,200,029.35	0.045	1,146,667.36	0.045	**645,856.63**	119.28
13	80	3	2,040,893.95	0.037	2,137,195.21	0.037	2,626,435.45	0.037	**993,491.35**	67.53
14	80	5	1579200.05	0.051	1,523,070.71	0.051	1,600,167.43	0.052	**986,667.36**	155.52
15	90	3	2,584,082.45	0.040	2,578,384.36	0.040	2,868,999.39	0.039	**1,417,865.08**	85.96
16	90	5	2,196,776.09	0.058	2,934,845.58	0.055	2,223,519.50	0.058	**1,385,903.12**	193.53
17	100	3	3,185,102.15	0.043	3,343,528.97	0.043	3,728,149.09	0.043	**1,724,489.90**	104.81
18	100	5	2,749,729.98	0.066	2,282,895.99	0.066	3,250,145.38	0.067	**1,512,187.47**	235.88

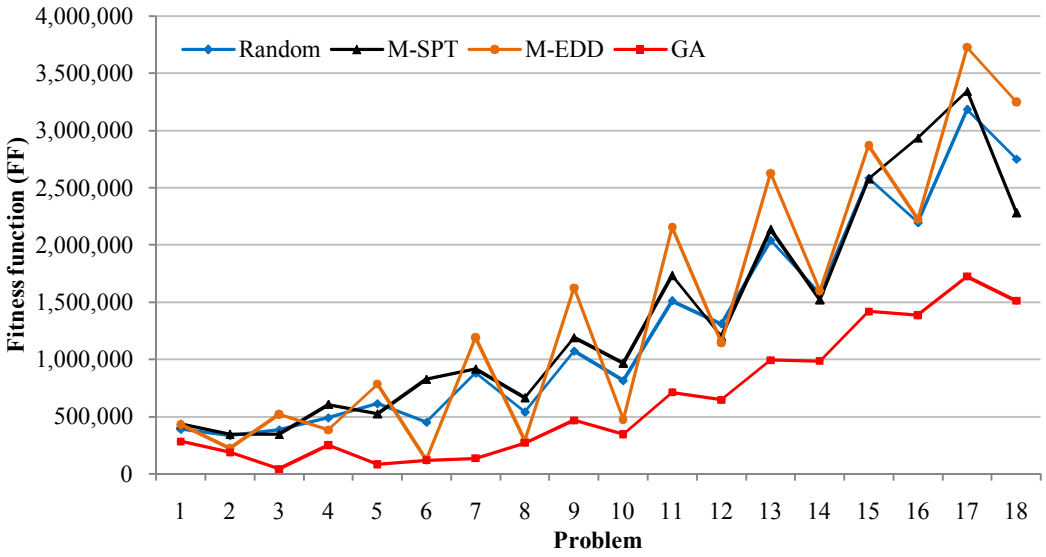

Fig. 5 Comparison of heuristics and GA for all large-sized problems

As depicted in Fig. 5, the resultant schedules from GA performed better in all cases.

Comparing effectiveness of solution methods

Total earliness and tardiness cost measure is used to evaluate the performance of the solution approaches. A single factor ANOVA method is used to find out whether there is a significant difference between performances of approaches. Table 4 shows ANOVA results for the solution algorithms. There is at least one algorithm with different response based on obtained results (p-value = 0.044) when confidence level is set at 0.95. Thus, Fisher's least significant difference method is applied to compare the performance of approaches. Table 5 presents the results of this method. The results confirm that there is a significant difference between GA with M-EDD, M-SPT and Random. Also, Table 5 shows no significant difference between M-EDD with M-SPT and Random and between M-SPT and Random.

Table 4 ANOVA results for solution methods

Source	df	SS	MS	F	P-value
Algorithm	3	7.09E+12	2.36E+12	2.84	0.044
Error	68	5.66E+13	8.32E+11		
Total	71	6.37E+13			

Table 5 Fisher 95% individual confidence intervals all pair-wise comparisons

Algorithm	Lower	Upper	Significant difference at 95% level
GA vs. M-EDD	175,502	1,389,086	Yes
GA vs. M-SPT	115,182	1,328,766	Yes
GA vs. Random	35,731	1,249,315	Yes
M-EDD vs. M-SPT	-667,111	546,473	No
M-EDD vs. Random	-746,563	467,021	No
M-SPT vs. Random	-686,243	527,341	No

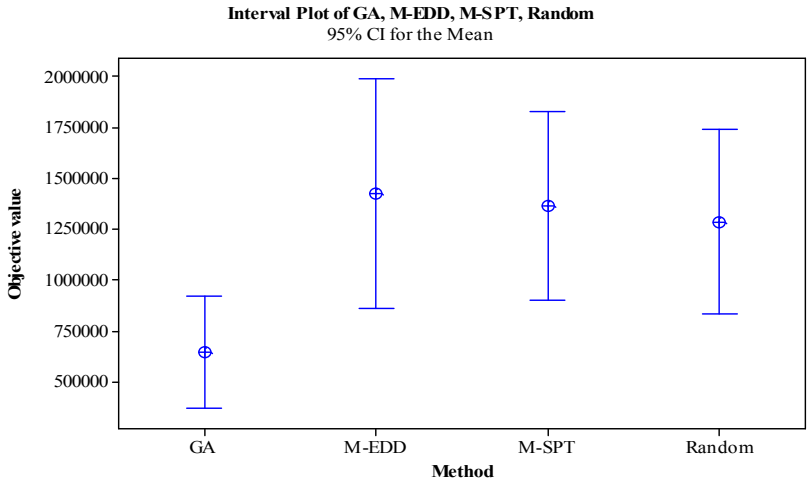

Fig. 6 Means and interval plot for objective value

As can be seen from Fig. 6, genetic algorithm for related objective function in studied problem is better than all of other heuristics. Due to the computational results, the superiority

of the proposed solution approach is concluded over other heuristics including M-EDD, M-SPT and Random.

5 Conclusions

The aim of this paper is to formulate a single machine scheduling problem with considering work and rework processes in manufacturing stages. A mixed-integer programming model is proposed for determining the optimum sequence of jobs to minimize the sum of earliness and tardiness costs. An adapted genetic algorithm is used to find the near optimal solution of the proposed model for large-sized problems. Computational experiments showed that the implemented meta-heuristic method can obtain good-quality solutions for all the test problem instances.

Future research may be interested in further improvement of the MIP models considering other assumptions such as sequence-dependent setup times, transportation constraints and availability constraints. Due to the complexity of problem, the computation time grows exponentially with problem size for exact algorithms, it is desired to develop a polynomial time heuristic in the further research. As another interesting line for further research other meta-heuristic algorithms with local search can be used to solve the problem.

References

1. Flapper, S.D.P., Fransoo, J.C., Broekmeulen, R.A.C.M. and Inderfurth, K. (2002). Planning and control of rework in the process industries: A review, Production Planning & Control, 1, 26–34.
2. Flapper, S.D.P. and Jensen, T. (2003). Logistic planning and control of rework, International Journal of Production Research, to appear.
3. Inderfurth, K. and Teunter, R.H. (2003). Production planning and control of closed-loop supply chains, In: Guide, Jr., V.D.R., van Wassenhove, L.N. (Eds.), Business Perspectives on Closed-Loop Supply Chains. Carnegie Mellon University Press, Oxford, 149–173.
4. Teunter, R.H. and Flapper S.D.P. (2003). Lot-sizing for a single-stage single-product production system with rework of perishable production defectives, OR Spectrum, 25, 85–96.
5. Haji, R., Haji, A., Sajadifar, M. and Zolfaghari, S. (2008). Lot sizing with non-zero setup times for rework, Journal of Systems Science and Systems Engineering, 17 (2), 230-240.
6. Inderfurth, K., Janiak, A., Kovalyov, M.Y. and Werner, F. (2006). Batching work and rework processes with limited deterioration of reworkables, Computers and Operations Research, 33 (6),1595-1605.
7. Inderfurth, K., Kovalyov, M.Y., Ng, C.T. and Werner, F. (2007). Cost minimizing scheduling of work and rework processes on a single facility under deterioration of reworkables, International Journal of Production Economics. 105 (2), 345-356.
8. Kuhl, M.E. and Laubisch, G.R. (2004). A simulation study of dispatching rules and rework strategies in semiconductor manufacturing, IEEE/SEMI advanced semiconductor manufacturing conference 4–6 May 2004, Boston, Massachusetts, USA, 325–329.
9. Sha, D.Y., Hsu, S.Y., Che, Z.H. and Chen, C.H. (2006). A dispatching rule for photolithography scheduling with an on-line rework strategy, Computers & Industrial Engineering, 50 (3), 233-247.
10. Shin, H.J. and Kang, Y.H. (2010). A rework-based dispatching algorithm for module process in TFT-LCD manufacture, International Journal of Production Research, 48 (3), 915-931.
11. Kang, Y.H. and Shin, H.J. (2010). An adaptive scheduling algorithm for a parallel machine problem with rework processes, International Journal of Production Research, 48 (1), 95-115.
12. Flapper, S.D.P. and Teunter, R.H. (2004). Logistic planning of rework with deteriorating work in process", International Journal of Production Economics, 88 (1), 51–59.
13. Inderfurth, K., Lindner, G. and Rachaniotis, N.P. (2005). Lot sizing in a production system with rework and product deterioration, International Journal of Production Research, 43 (7), 1355–1374.
14. Barketau, M.S., Cheng, T.C.E. and Kovalyov, M.Y. (2008). Batch scheduling of deteriorating reworkables, European Journal of Operational Research, 189 (3), pp. 1317-1326.

15. Tayebi Araghi, M.E., Jolai, F., and Rabiee, M. (2014). Incorporating learning effect and deterioration for solving a SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach, International Journal of Computer Integrated Manufacturing, 27(8), 33-746.
16. Yin, Y., Wu, W-H., Cheng, T.C.E., and Wu, C-C. (2015). Single-machine scheduling with time-dependent and position-dependent deteriorating jobs, International Journal of Computer Integrated Manufacturing, 28(7), 781-790.
17. Mahnam, M., Moslehi, G., and Fatemi Ghomi. S.M.T. (2013). Single Machine Scheduling with Unequal Release Dates and Idle Insert for Minimizing the Sum of Maximum Earliness and Tardiness, Mathematical and Computer Modelling, 57, 2549–2563.
18. Keyvanfar, V., Mahdavi, I., and Komaki. G.H.M. (2013). Single Machine Scheduling with Controllable Processing times to Minimize Total Tardiness and Earliness, Computers & Industrial Engineering, 65, 166–175.
19. Seidgar, H., Kiani, M., Abedi, M., and Fazlollahtabar. H. (2014). An Efficient Imperialist Competitive Algorithm for Scheduling in the Two-stage Assembly Flow Shop Problem, International Journal of Production Research, 52(4) 1240–1256.
20. Seidgar, H., Abedi, M., and Tadayonirad S. (2015). A hybrid particle swarm optimisation for scheduling just-in-time single machine with preemption, machine idle time and unequal release times, International Journal of Production Research, 53(6), 1912-1935.
21. Ramezanian, R. and Saidi-Mehrabad, M., (2012). Multi-product unrelated parallel machines scheduling problem with rework processes, Scientia Iranica E, 19(6), 1887–1893.
22. Gen, M., Tsujimura, Y. and Kubota, E. (1994). Solving job-shop scheduling problem using genetic algorithms, In Proceeding of the 16th international conference on computer and industrial engineering, Ashikaga, Japan, 576–579.
23. Gao, J., Gen, M., Sun, L. and Zhao, X. (2007). A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems, Computers & Industrial Engineering, 53, 149–162.
24. Gen, M. and Cheng, R. (1997). Genetic algorithms & engineering design, New York: Wiley.