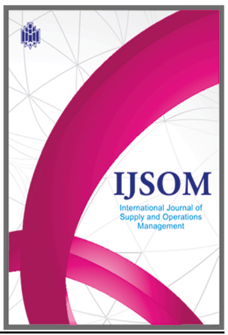
International Journal of Supply and Operations Management

IJSOM

**November 2016, Volume 3, Issue 3, pp. 1413- 1428**
**ISSN-Print: 2383-1359**
**ISSN-Online: 2383-2525**
**www.ijsom.com**



# A Scheduling Model for the Re-entrant Manufacturing System and its Optimization by NSGA-II

Masoud Rabbani [a,*], Safoura Famil Alamdar[a], Parisa Famil Alamdar[b]

[a] *School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran*
[b] *School of Industrial Engineering, Amir Kabir University, Tehran, Iran*

## Abstract

In this study, a two-objective mixed-integer linear programming model (MILP) for multi-product re-entrant flow shop scheduling problem has been designed. As a result, two objectives are considered. One of them is maximization of the production rate and the other is the minimization of processing time. The system has $m$ stations and can process several products in a moment. The re-entrant flow shop scheduling problem is well known as NP-hard problem and its complexity has been discussed by several researchers. Given that NSGA-II algorithm is one of the strongest and most applicable algorithm in solving multi-objective optimization problems, it is used to solve this problem. To increase algorithm performance, Taguchi technique is used to design experiments for algorithm's parameters. Numerical experiments are proposed to show the efficiency and effectiveness of the model. Finally, the results of NSGA-II are compared with SPEA2 algorithm (Strength Pareto Evolutionary Algorithm 2). The experimental results show that the proposed algorithm performs significantly better than the SPEA2.

**Keywords:** Re-entrant Manufacturing System, Non-dominated Sorting Genetic Algorithm (NSGA-II), Taguchi Parameter Setting.

*corresponding Author email address: mrabani@ut.ac.ir

## 1. Introduction

In the classical flow shop scheduling model, a usual assumption is that each job is processed on each machine at most once. Nevertheless, in a re-entrant flow shop system, this assumption does not exist. The re-entrant manufacturing system detects that paths of all jobs are identical, while some jobs will process on some machines more than once at different stages of their processing. A typical example of the re-entrant manufacturing systems is the semiconductor manufacturing system. The re-entrant flow shop scheduling is NP-hard problem and its complexity has been discussed by several researchers (Cavory et al., 2005; Hsu et al., 2008; Kubale and Nadolski, 2005). Heuristic and Meta-heuristic algorithms, especially genetic algorithm (GA) were employed to solve the re-entrant scheduling problem (Hwang and Sun, 1998; Lee and Lin, 2010; Rau and Cho, 2009; Lin et al., 2013). Kim et al. (2006) considered a re-entrant system with either a single-job machine or a batch machine at each workstation and estimated the performance of this system with analytical method, based on mean value analysis approximation and heuristic adjustments.

Choi & Kim (2008) considered an m-machine re-entrant flow shop scheduling problem. The objective of this model is the minimization of makespan. In this paper, paths of all jobs are identical as in ordinary flow shops, but the jobs are processed multiple times on the machines. They developed several types of heuristic algorithms, and compared solutions of these algorithms with lower bounds as well as solutions obtained from SA algorithms.

Jing et al. (2008) examined a two-machine re-entrant flow shop scheduling problem with the objective of minimizing makespan. They suggested a number of heuristics and evaluated their efficiency via computational experiments. Chen et al. (2008) considered RPFS scheduling, and applied hybrid tabu search (HTS) to minimize the makespan of jobs. The hybridization method was utilized to enhance pure tabu search (TS) performance. Choi & KO (2009) developed a simulation-based two-phase genetic algorithm for the capacitated re-entrant scheduling problem. A job priority-based randomized policy was described, showing the job priority and the properness of local non-idling in appropriation buffering as well as the process capacity to available job cases. Choi & Kim (2009) presented a branch and bound algorithm for a two-machine re-entrant flow shop scheduling problem with the objective of minimizing total tardiness. In this re-entrant flow shop, all jobs must be processed on machine 1, machine 2 and subsequently on machine 1 and 2. They developed dominant properties, a lower bound and heuristic algorithms for this problem, and developed a branch and bound algorithm.

Fattahi et al. (2010) studied the problem of the single-product re-entrant flexible manufacturing system scheduling, in which maximizing the production rate is considered as the main objective and minimizing the work in process is considered as the second objective. An algorithm based on the simulated annealing algorithm was developed to solve the medium and large size problems. Jing et al. (2011) studied the re-entrant flow shop scheduling problem with the criteria of minimizing total completion time. A heuristic and its variations were proposed to solve the problem, by first developing a k-insertion method. Kumar et al. (2012) considered the machine loading problem in flexible manufacturing system with objectives of minimization of system unbalance and maximization of throughput. A meta-hybrid heuristic technique based on genetic algorithm and particle swarm optimization was used to solve this problem.

Dong & He (2012) proposed single-product re-entrant manufacturing systems and modeled the basic partial differential equation for this problem. The numerical examples implied that the basic continuous models did not perform well for single-product re-entrant systems. A new state equation which takes into consideration the degree of a product is presented to improve the basic continuous models. Lin et al. (2013) considered the re-entrant flow shop scheduling problem (RFSP) with dual resource constraints. For balancing the rework percentage and causing higher resource consumption and minimizing the makespan, a multi-level genetic algorithm was proposed. Xu et al. (2014) considered the re-entrant flow shop scheduling problem to minimize the makespan. They adopted the CPLEX solver and utilized a memetic algorithm (MA) to solve this problem. Jeong & Kim (2014) considered re-entrant flow shop scheduling problem with two machines. In this system, all jobs must be processed twice on each machine with sequence-dependent setup times on the second machine. The objective is the minimization of total tardiness with developed dominance properties and a lower bound by extending those for a two-machine re-entrant flow shop system, and also presenting a branch and bound algorithm in which these dominance properties, lower bound, and heuristics were utilized.

Mirabi et al. (2014) developed a hybrid genetic algorithm to minimize the holding, delay and setup costs of large permutation flow-shop scheduling problems with sequence-dependent setup times on each machine. Hinze (2015) presented a lot streaming formulation for a re-entrant flow shop scheduling system with missing operations. Two objectives were considered: the makespan and the sum of completion times ere compared to an existing approach.

Xu et al. (2016) considered re-entrant flow shop scheduling with a position-based learning effect to minimize the total completion time. They developed a Heuristic based genetic algorithms to solve this problem. Belkaid et al. (2016) studied a scheduling problem for reentrant parallel machines with consumable resources. The resources availability, jobs assignment and sequencing at each cycle were considered and a genetic algorithm was developed to minimize the makespan of the considered problem. Kia et al. (2017) studied a dynamic flexible flow line problem with sequence-dependent setup times. Minimization of mean flow time and mean tardiness was considered and four composite dispatching rules were proposed to solve it by applying genetic programming framework.

According to aforementioned papers, the contribution of this paper is the presentation of a scheduling model for the multi-product re-entrant manufacturing system which has $m$ stations and uses non-dominated genetic algorithm (NSGA-II) for solving the two-objective problem in flow shop system. In this paper, due to problem assumptions, a mathematical model for m-machine re-entrant flow shop scheduling problem is designed. Two objectives are considered for this problem. One of them is the maximization of production rate and another is the minimization of process time which indicates time interval between start time of the first operation of a work in the system and finish time of the last operation of that work in the system. This objective will lead to reduction in inventory and thereby achieving goals of just-in-time production. Problem includes scheduling $N$ kinds of work (product or piece) which is processed by a multi-stage system which has $m$ stations. Since the problem is known as NP-hard problem, NSGA-II is developed to solve the problem. The rest of the paper is organized as follows: In the next section, Problem formulation, the proposed mathematical model and notation as well as assumptions are proposed, in Section3, non-dominated genetic algorithm (NSGA-II) is developed for solving the problem. Computational experiments are carried out for evaluating the performance of the suggested Metaheuristic and results are given in Section 4. In Section 5, the results of the proposed algorithm are compared with SPEA2 algorithm (Strength Pareto Evolutionary Algorithm 2) to validate the performance of it. Finally, Section 6 includes concluding remarks.

## 2. Problem description

Flexible production system studied in this research has m-stations or facilities and each one includes equipment to carry out variety of operations. These facilities are fixed and transportation time between them has been considered trivial. This system has received pieces/works/product orders in the high volume and has coordinated system to produce them. In each step, the system has been designed in such a way that it provides conditions and advantages of mass production in producing one piece. The considered problem encompasses scheduling $N$ kinds of work (product or piece) which is processed by a multi-stage system which has $m$ production stations. Adequate number of works has formed a shipment and shipments are being nourished one after the other to the system. To complete works, a set of operations in the form of a specified order (production route) has been considered and each operation can be performed by one station. Due to the considered production system (re-entrant manufacturing system) in this paper, jobs will visit certain machines more than once at different stages of processing. We use the following assumptions and notation in this paper: We have $N$ diverse of products and each product is produced in high circulation.This implies that, from each kind of product $u(u = 1,2, \dots, N), n_u$ similar products (work) are produced in one cycle $\{n_1, n_2, \dots, n_N\}$. This problem has $m$ stations which show that $M = \{M_1, M_2, \dots, M_m\}$. Based on the problem assumption in which adequate number of one work (piece, product or a set of operations) was demanded; operations of some works have been programmed as a cycle so that they can form a simple cycle with high efficiency and can be regularly repeated. Production operation of one work can't be stopped and one station is not allocated to another work after starting an operation until the end of the respective operation. The considered work has $H$ operation and each operation has a processing time equal to to $P_{uh}$ which is fixed and has been previously specified. $H_u$ Denotes the number of operations for product $u(H_1, H_2, \dots H_N)$ and for each product $h = 1,2, \dots H_u(u = 1,2, \dots, N)$.To produce similar products related to one product, similar operation is required. That is, $H_u = H_{uj}$. Production facilities which have the ability to perform one operation have been previously determined. The predetermined facility for each operation is defined by a binary parameter $a_{iuh}$ as described below:

$$a_{iuh} = \begin{cases} 1 & \text{if operation h from work u can be done by facility i} \\ 0 & \text{otherwise} \end{cases}$$

Start time of operation $h$ from product $u$ (for similar $j^{th}$ work) is shown with $S_{ujh}$ and finish time of operation $h$ from product $u$ (for similar jth work) is shown with $F_{ujh}$.The operation $h$ from the work $u$ for the similar work $j$ is denoted by $O_{ujh}$ and the total number of operations allocated to the machine $i$ is denoted by $K_i$. In order to display the sequence of operations in one station, a priority factor has been considered for each station. This priority factor exhibits the order of various operations which have to be carried out by one station in one cycle. This sequence is defined by a binary variable $x_{iujnk}$. If facility $i$ executed operation $h$ of product $u$ for similar work $j$ in priority $k$ then $x_{iujnk}$ is equal 1.

$$x_{iujnk} = \begin{cases} 1 & \text{if operation } h \text{ from product } u \text{ for similar work } j \text{ over the machine } i \\ & \text{is being done in priority } k \\ 0 & \text{otherwise} \end{cases}$$

Start time of station $i$ in the work priority $k$ is shown with $Tm_{i,k}$ and finish time of station $i$ in the work priority $k$ is shown with $Fm_{i,k}$. As earlier mentioned, one of the objective functions of the problem is to increase the production rate. Because increase in production rate is limited to bottleneck stations, minimization of the idle time for the bottleneck was regarded as an objective.

Due to problem condition, it can be articulated that the objective of maximizing production rate is equivalent to the objective function of minimizing workload of a station which has the maximum workload. Therefore, this objective function is calculated as follows:

$$load_i = Fm_{i,ki} - Tm_{i,k} \qquad i = 1,2,\dots,m$$
$$Z_1 = min\left(\frac{1}{n_1 + n_2 + \cdots n_N} \; max\{loud_i \;\; , i = 1,2,, \dots, m\}\right)$$

$load_i$ Shows the working load of station $i$ in one cycle. This rate is obtained from subtracting the start time of the station from the finish time of the last operation of this station in one cycle. Where $max\{loud_i \;\; , i = 1,2,, \dots, m\}$ defines the working load of the bottleneck station. Working load rate determines the bottleneck station. Regarding the issue that a lot of works are processed in one cycle, in order to calculate cycle time, the obtained rate is divided into number of works in one cycle and cycle time of one piece is calculated. Because one of the important goals in mass production is to minimize flow time and work in process (WIP), the second target function was considered as the decrease in WIP rate. This target function is considered as the flow time of all works in one cycle which is calculated as follows. This target specifies sequence of operations over each machine in cycle scheduling (by minimizing WIP).

$$FF_{uj} = F_{ujH_u} - S_{uj1}$$
$$\overline{FF_u} = \frac{1}{n_u}\sum_{j=1}^{n_u} F_{uj}$$
$$Z_2 = min(\overline{FF_1} + \overline{FF_2} + \cdots + \overline{FF_u} + \cdots + \overline{FF_N}) = \min\left(\sum_{u=1}^{N} \overline{FF_u}\right)$$

Where $FF_{uj}$ is the time interval between start time of the first operation of work $j$ and the end of the last operation (flow time of similar work $j$ from product u). $\overline{FF_u}$ Shows the average flow time of one cycle for all of the similar works of product $u$.

### *2.1. The proposed mathematical model*

According to earlier mentioned assumptions, mixed-integer linear programming for solving two-objective re-entrant flexible production scheduling problem is as follows:

$$Z_1 = min\left(\frac{1}{n_1 + n_2 + \cdots n_N} \; max\{loud_i \;\; , i = 1,2,, \dots, m\}\right) \tag{1}$$

$$Z_2 = min(\overline{FF_1} + \overline{FF_2} + \cdots + \overline{FF_u} + \cdots + \overline{FF_N}) = \min\left(\sum_{u=1}^{N} \overline{FF_u}\right) \tag{2}$$

s.t.

$$S_{ujh} + P_{uh} \le S_{u,j+i,h} \qquad for \begin{cases} u = 1,2,\dots,N \\ j = 1,2,\dots,(n_u - 1) \\ h = 1,2,\dots,H_u \end{cases} \tag{3}$$

$$S_{ujh} + P_{uh} \le S_{u,j,h+1} \qquad for \begin{cases} u = 1,2,\dots,N \\ j = 1,2,\dots,n_u \\ h = 1,2,\dots,(H_u - 1) \end{cases} \tag{4}$$

$$Tm_{i,k} + P_{uh}\, x_{iujhk} \le Tm_{i,k+1} \qquad for \begin{cases} i = 1,2,\dots,m \\ k = 1,2,\dots,(k_i - 1) \\ u = 1,2,\dots,N \\ j = 1,2,\dots,n_u \\ h = 1,2,\dots,H_u \end{cases} \tag{5}$$

$$Tm_{i,k} \leq S_{ujh} + (1 - x_{iujhk})Q \qquad for \begin{cases} i = 1,2,\dots,m \\ k = 1,2,\dots,K_i \\ u = 1,2,\dots,N \\ j = 1,2,\dots,n_u \\ H = 1,2,\dots,H_u \end{cases} \tag{6}$$

$$Tm_{i,k} + (1 - x_{iujhk})Q \geq S_{ujh} \qquad for \begin{cases} i = 1,2,\dots,m \\ k = 1,2,\dots,K_i \\ u = 1,2,\dots,N \\ j = 1,2,\dots,n_u \\ H = 1,2,\dots,H_u \end{cases} \tag{7}$$

$$\sum_u x_{iujhk} \leq a_{iuh} \qquad for \begin{cases} i = 1,2,\dots,m \\ k = 1,2,\dots,K_i \\ j = 1,2,\dots,n_u \\ H = 1,2,\dots,H_u \end{cases} \tag{8}$$

$$Fm_{i,k} + (1 - x_{iujhk})Q \geq F_{ujh} \qquad for \begin{cases} i = 1,2,\dots,m \\ k = 1,2,\dots,K_i \\ u = 1,2,\dots,N \\ j = 1,2,\dots,n_u \\ H = 1,2,\dots,H_u \end{cases} \tag{9}$$

$$load_i = Fm_{i,ki} - Tm_{i,k} i = 1,2,\dots,m \tag{10}$$

$$FF_{uj} = Fu_i H_u - S_{i,k} i = 1,2,\dots,m \tag{11}$$

$$\overline{FF}_u = \frac{1}{n_u} \sum_{j=1}^{n_u} F_{uj} \tag{12}$$

$$\sum_u \sum_j \sum_h x_{iujhk} = 1 \qquad for \;\; i = 1,2,\dots m \quad , \quad = 1,2,\dots,k_i \tag{13}$$

$$\sum_i \sum_k x_{iujhk} = 1 \qquad for \begin{cases} u = 1,2,\dots,N \\ j = 1,2,\dots,n_u \\ h = 1,2,\dots,H_u \end{cases} \tag{14}$$

$$S_{ujh} \geq 0 \text{ ، } F_{ujh} \geq 0 \;, u = 1,2,\dots,N \;, \; i = 1,2,\dots,n_u \;, h = 1,2,\dots,H_u \tag{15}$$

$$Tm_{i,k} \geq 0 \text{ ، } Fm_{i,k} \geq 0 \qquad i = 1,2,\dots,m \qquad k = 1,2,\dots,k_i \tag{16}$$

$$load_i \geq 0 \text{ ، } FF_{uj} \geq 0 \;, \; u = 1,2,\dots,N \;, j = 1,2,\dots,n_u \;, i = 1,2,\dots,m \tag{17}$$

$$x_{iuhhk} \epsilon \{0,1\} \qquad \begin{cases} i = 1,2,\dots,m \\ u = 1,2,\dots,N \\ j = 1,2,\dots,n_u \\ H = 1,2,\dots,H_u \\ k = 1,2,\dots,k_i \end{cases} \tag{18}$$

Functions (1) and (2) specifies objective functions of the problem. Constraints (3) and (4) display prerequisite relations of operations. Constraints (5), (6) and (7) will cause respective constraints related to the possibility to perform one operation in one moment over one station and also relations related to the stations priorities will be met. Constraint (8) shows that the station needed for each operation will be chosen among stations allocated to that operation. Constraints (9) to (12) show equations related to the objectives. Constraints (13) and (14) ensure that only one priority is assigned to each operation and only one operation is assigned to each priority. The other Constraints are non-negative and binary Constraints.

## 3. Proposed Meta Heuristic algorithm

Deb et al. (2002) developed a fast and elitist multi-objective non-dominated sorting genetic algorithm without selecting the sharing parameters. They called this approach “non-dominated genetic algorithm version 2 (NSGA-II)”. NSGA-II is one of the most efficient and well-known multi-objective evolutionary algorithms and has been successfully utilized in several production systems and shows that it is a very promising and efficient optimization technique. This algorithm applies the fast non-dominated sorting technique and a crowding distance to rank and choose the population fronts. Thereafter, the algorithm uses the standard bimodal crossover and polynomial operators to incorporate the current population and its offspring generated as next generation. Finally, in terms of non-dominance and diversity, the best individuals are chosen as the solutions. In the following, specifications of the proposed algorithm will be explained.

### *3.1. Chromosome definition*

In this algorithm, the length of the chromosome was considered equal to number of works, which means total number of states. For example, if the number of products is 3 and from product 1, 2 and 3, 1, 3 and 2 kinds of products is needed respectively, and the number of their operations is 4, 5 and 6, respectively; then total number of states is 31.Chromosome length will also be equivalent to 31 where the first row of chromosome shows the operation sequence (state number) while the second row shows the machine’s number.

### *3.2. Non-dominated sorting*

Before the selection is done, a rank based on non-domination is assigned to each person (chromosome) in the population and all of non-dominated chromosomes are classified into one category (with a dummy fitness value which is consistent with population size). Also, crowding distance is being calculated using the following formula (Eq. (19) below) so as to keep a diverse front by ensuring that each member stays a crowding distance apart. This issue helps algorithm to keep the population diverse, and helps the algorithm to explore the fitness landscape. Non-dominated sorting is performed both after crossover and mutation. In sorting, solutions are ranged based on crowding distance from big to small and arranged from small to big base on Pareto front. This matter causes solutions which are better in terms of Pareto front, to be ranked higher and those that are better in this front in terms of crowding distance are ranked higher. First selection criteria in algorithm are solution rank and secondly, crowding distance related to the solution. The less solution rank and greater crowding distance is more desirable.

$$cd_i = \sum_{j\in\{F_{f_i}\}, i\neq j} d_{ij}$$

$$d_{ij} = \sqrt{\sum_{m=1}^{2} \left(\frac{f_m(i)-f_m(j)}{max\{f_m(0)\}-min\{f_m(0)\}}\right)^2}$$

*(19)*

Where $cd_i$ is crowding distance for individual (chromosome) $i$; $f_m(i)$ is the value of objective of $i^{th}$ individual; $d_{ij}$ is crowding distance of individual $i$ and $j$.

### *3.3. Crossover*

For crossover operation to be performed, the individuals are selected via Tournament selection method. In this algorithm, 2 crossover methods have been considered, in which algorithm selects one of them randomly in each iteration.

1. One-point crossover: one column (gene) is chosen from chromosome. The first sector from the first parent and the second sector from the second parent create the first offspring. Also, the second offspring is created from the second sector of the first parent and the first sector of the second parent.

2. Two-point crossover: 2 genes are randomly chosen and values between these two columns are taken from another parent.

### *3.4. Mutation*

Selecting parent is performed based on tournament selection. Here, 2 mutations have been regarded whereby algorithm chooses one of them randomly in each iteration.

Swap mutation: Two genes are randomly chosen and their values are exchanged. Most times, this method keeps information related to the neighborhood (4 junctions are broken down) and will cause more damage in genes' information.
Inversion mutation: Two genes are randomly chosen and substring between them is reversed. Most times, this method retains information related to the neighborhood (only 2 junctions are broken down), but it is destructive of ordering information.

### *3.5. Modifying prerequisite relations of activities*

Prerequisite of any state is the previous state of that product from its kind. That is, it considers its kind as a prerequisite, because it is not supposed to produce products one after another and parallel state is also considered. By this definition, it allows the algorithm to choose the optimal state. Modifying prerequisites has been carried out over the created initial random solution based on RCPSP.

### *3.6. Convergence and termination of Algorithm*

The last step in NSGA-II algorithm is to investigate termination condition. There are several methods such as algorithm repetition, lack of enhancement in solutions after multiple repetitions, algorithm right implementation time and combination of mentioned methods. In this paper, due to parameters setting in the design of experiments, defined iteration of algorithm was employed for termination.

## 4. Experimental results

In order to set parameters of algorithm, there are several statistical methods to design experiments. In this research, Taguchi method was employed to determine the most influenced factor for NSGA-II and the best combination of parameters. Taguchi (1999) enhanced a family of matrixes of trivial factorial experiments, so that he could perform an experiment design so as to reduce number of experiments for one problem. According to the literature and initial tests, the levels for parameters of the algorithm including crossover probability (Pc) (0.95, 0.9, and 0.8); mutation probability (Pm) (0.01, 0.05, and 0.1); population size (200, 100, and 50) and maximum iteration (150, 100, and 50) are considered for Taguchi experiment.
Based on this method, 9 different scenarios are considered. Six instances dissolved and each instance has been run five times under different scenarios. Taguchi method is designed to maximize the performance criterion which is called the signal-to-interference ratio (SNR or S/N). Key variables affecting the qualitative characteristics of process can be identified by this criterion. SRN is obtained from the equation (20):

$$\mathrm{SRN} = -10 log_{10}\left(\frac{\sum y_i{}^2}{n}\right) \tag{20}$$

Where $y_i$ Shows the response variable of the $i-th$ observation and $n$ is the number of experiments. In addition to SNR, the Percent relative deviation (PRD) is used which is calculated as follows:

$$PRD = \frac{OF_i - OF_{min}}{OF_{max} - OF_{min}} \quad (21)$$

Where $OF_i$ is the obtained value in the experiment *i*, $OF_{min}$ is the minimum value of observations in all experiment and $OF_{max}$ is the maximum value of observations in all experiment.
After performing a series of experiments, results obtained from all implementations of algorithms by Taguchi design of experiments are presented in Table1 and Figures 1-2.

**Table1.** Design of appropriate parameters

| | | level 1 | level 2 | level 3 |
|---|---|---|---|---|
| S/N | Pc | -6.805275627 | -6.825406682 | -6.825162433 |
| | Pm | -6.792539999 | -6.8173718 | -6.845932944 |
| | popsize | -9.875417335 | -6.767184304 | -3.813243104 |
| | Maxiter | -10.47750098 | -6.511563337 | -3.466780427 |
| RPD | Pc | 0.49769702 | 0.517322066 | 0.483566513 |
| | Pm | 0.498312751 | 0.504800737 | 0.495472111 |
| | popsize | 0.534481932 | 0.468882959 | 0.495220708 |
| | Maxiter | 0.519375818 | 0.500080564 | 0.479129217 |

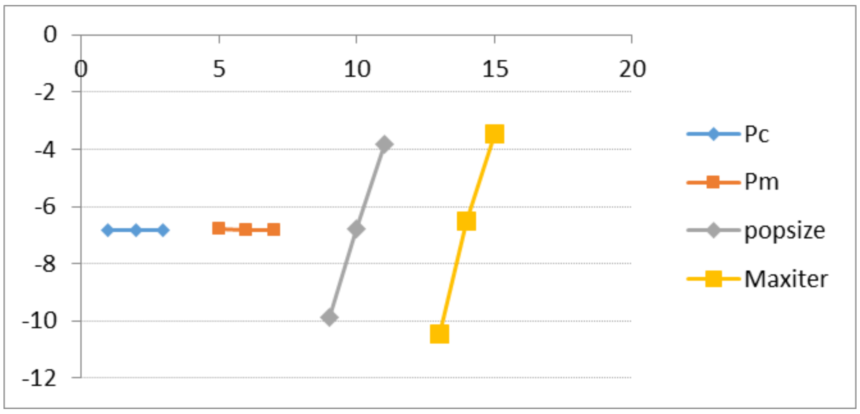


**Figure 1.** Means plot of S/N for NSGA-II parameters.

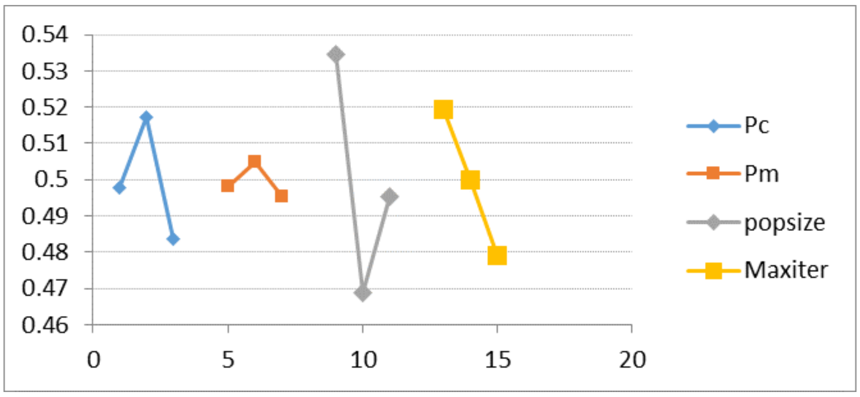


**Figure 2.** Means plot of RPD for NSGA-II parameters.

According to the above results, the optimal values for parameters of algorithm is selected as 0.8 for Pc, 0.1 for Pm, 100 for population size and 50 for maximum iteration.
Then, four problems by different sizes have been solved so as to demonstrate algorithm efficiency. The experiments were run on a PC with 2.50 GHz processor and 8.00 GB memory. Matlab was used to program the NSGA-II algorithm. In this algorithm, initial data were randomly produced. Characteristics of these four problems are shown in Table 2. Also in Table 3, data related to the second problem was exhibited as the sample. The results related to solving the aforementioned problems are displayed in Tables 4-5.

**Table2.** Characteristics of studied problem

| Problem | Number of products | Number of product's kinds | Number of operations | Size |
|---|---|---|---|---|
| 1 | 3 | }2 3 1{ | }6 5 4{ | 31 |
| 2 | 8 | }5 7 4 3 2 5 7 7{ | }4 5 4 5 3 5 4 6{ | 187 |
| 3 | 5 | }10 9 8 9 7{ | }6 6 6 5 5{ | 242 |
| 4 | 7 | }9 6 8 7 9 5 6{ | }5 5 4 4 5 3 6{ | 231 |

**Table3.** Data of second problem

| Products | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Number of kinds | | 5 | 7 | 4 | 3 | 2 | 5 | 7 | 7 |
| Number of operations | | 4 | 5 | 4 | 5 | 3 | 5 | 4 | 6 |
| Time of operations | 1 | 23 | 99 | 95 | 25 | 38 | 14 | 53 | 10 |
| | 2 | 42 | 85 | 3 | 17 | 43 | 82 | 47 | 15 |
| | 3 | 12 | 57 | 46 | 60 | 72 | 61 | 26 | 25 |
| | 4 | 61 | 28 | 18 | 51 | | 21 | 70 | 77 |
| | 5 | | 39 | | 32 | | 40 | | 70 |
| | 6 | | | | | | | | 20 |

**Table3.** Continued

| Products | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Pre-determined facility for each operation | 1 | 2,4 | 1,2,3,4,5 | 1,2,3,4,5 | 1,2 | 1,2,5 | 1,2 | 1,2,4 | 2,4 |
| | 2 | 1,3,5 | 1,2,4,5 | 1,2,4,5 | 2,4,5 | 1,4,5 | 2,3,4 | 1,5 | 1,2,5 |
| | 3 | 1,2,3,5 | 1,3,4 | 1,3,4 | 1,3,5 | 1,4 | 1,3,5 | 1,2,3 | 1,2,3 |
| | 4 | 5 | 3,5 | 3,5 | 3,4 | | 2,4 | 2,3,5 | 2,3,5 |
| | 5 | | 3,4,5 | | 1,2,4,5 | | 1,4,5 | | 1,2,3,4 |
| | 6 | | | | | | | | 3,4,5 |

**Table4.** The first 10s Pareto solutions obtained by NSGA-II

| Problem1 | | | Problem2 | | |
|---|---|---|---|---|---|
| Individual | Objective function values | | Individual | Objective function values | |
| | 1 | 2 | | 1 | 2 |
| 1 | 21.3548 | 743 | 1 | 18.6631016 | 2560.566667 |
| 2 | 15.2258 | 990.1667 | 2 | 14.15508021 | 3304.469048 |
| 3 | 16 | 863.5000 | 3 | 14.73262032 | 3186.542857 |
| 4 | 15.3548 | 919.6667 | 4 | 15.57219251 | 2974.047619 |
| 5 | 17.6774 | 815 | 5 | 14.97860963 | 3065.77619 |
| 6 | 18.7742 | 784 | 6 | 16.11229947 | 2901.695238 |
| 7 | 21.1935 | 768.6667 | 7 | 17.52406417 | 2721.669048 |
| 8 | 18.1613 | 796.8333 | 8 | 16.30481283 | 2843.130952 |
| 9 | 21.3548 | 743 | 9 | 18.2459893 | 2575.209524 |
| 10 | 18.4516 | 770 | 10 | 16.98395722 | 2816.4 |

**Table5.** The first 10s Pareto solutions obtained by NSGA-II

| Problem3 | | | Problem4 | | |
|---|---|---|---|---|---|
| Individual | Objective function values | | Individual | Objective function values | |
| | 1 | 2 | | 1 | 2 |
| 1 | 24.53719008 | 2092.249206 | 1 | 13.41558442 | 4370.060317 |
| 2 | 17.0661157 | 7172.69127 | 2 | 18.12121212 | 2258.03254 |
| 3 | 19.89669421 | 4165.767857 | 3 | 14.22510823 | 3248.854365 |
| 4 | 22.92975207 | 2256.383333 | 4 | 13.98701299 | 4095.699603 |
| 5 | 18.18181818 | 5323.776587 | 5 | 16.11255411 | 2347.519444 |

**Table5.** Continued

| Problem3 | | | Problem4 | | |
|---|---|---|---|---|---|
| Individual | Objective function values | | Individual | Objective function values | |
| | 1 | 2 | | 1 | 2 |
| 6 | 18.16115702 | 6752.756349 | 6 | 15.62770563 | 2471.890873 |
| 7 | 20.23140496 | 3268.79881 | 7 | 15.30735931 | 2982.451984 |
| 8 | 22.23140496 | 3156.58373 | 8 | 15.08225108 | 3242.521032 |
| 9 | 22.49586777 | 2418.305159 | 9 | 13.53679654 | 4199.328571 |
| 10 | 18.00413223 | 7147.693254 | 10 | 15.78354978 | 2433.884524 |

## 5. Empirical analysis

In this section, the results of the NSGA-II are compared with SPEA2 (Strength Pareto Evolutionary Algorithm version 2) according to Zitzler et al. (2001). This is an evolutionary algorithm with a Pareto Archive mechanism completed by a specific feature which keeps variety through a minimal distance between solutions. The non-dominated solutions fill the Pareto Archive (PA) until a given number $N$ of solutions are obtained. If there are no sufficient non-dominated solutions, the PA will be filled with dominated solutions which have the best objectives. When there are too many solutions in the PA, they are maintained in decreasing order of the distance $\sigma$ (Zitzler et al., 2001). The parameters of SPEA2 are similar to the parameters of NSGA-II algorithm with two extra ones: the size $N$ of the archive and the rank $\sigma$ of the comparison of each individual. The best parameters of this algorithm are defined as shown in Table 6.

**Table6.** Parameters for the SPEA2 algorithm

| | | |
|---|---|---|
| Population size=100 | Crossover prob.=0.7 | $\sigma$=7 |
| Mutation prob. = 0.1 | Archive size=50 | |

Table 7 compares the minimum and average values of the Pareto optimal solutions of two algorithms for the crossover probability of 0.5, 0.7 and 0.9. Table 8 shows the minimum and average values for Mutation probability 0.1, 0.3 and 0.5. Subsequently, the minimum and average values of two algorithms for Maxiter are shown in Tables 9–10. All these tables show significantly better results for the NSGA-II than SPEA2 algorithm.

**Table7.** Comparison of minimum and average values of two algorithms

| Crossover prob | Algorithms | | Objective 1 | Objective 2 |
|---|---|---|---|---|
| 0.5 | NSGA-II | Minimum | 15.2994 | 2786.288 |
| | | Average | 18.3816 | 3862.484 |
| | SPEA2 | Minimum | 18.1424 | 2964.122 |
| | | Average | 20.5413 | 3720.542 |
| 0.7 | NSGA-II | Minimum | 13.58289 | 2744.762 |
| | | Average | 16.14251 | 3373.973 |
| | SPEA2 | Minimum | 17.010 | 3012.542 |
| | | Average | 19.867 | 3460.850 |
| 0.9 | NSGA-II | Minimum | 13.19786 | 2644.433 |
| | | Average | 16.94652 | 3737.643 |
| | SPEA2 | Minimum | 17.862 | 3185.121 |
| | | Average | 24.152 | 3650.462 |

**Table 8.** Comparison of minimum and average values of two algorithms

| Mutation prob. | Algorithms | | Objective 1 | Objective 2 |
|---|---|---|---|---|
| 0.1 | NSGA-II | Minimum | 15.93048 | 3155.214 |
| | | Average | 19.10255 | 3446.046 |
| | SPEA2 | Minimum | 14.5317 | 3013.5412 |
| | | Average | 21.0214 | 3652.145 |
| 0.3 | NSGA-II | Minimum | 13.58289 | 2744.762 |
| | | Average | 16.14251 | 3373.973 |
| | SPEA2 | Minimum | 18.4512 | 3156.854 |
| | | Average | 20.3675 | 3890.20 |
| 0.5 | NSGA-II | Minimum | 14.54011 | 3505.7 |
| | | Average | 16.58671 | 3879.579 |
| | SPEA2 | Minimum | 16.980 | 3712.564 |
| | | Average | 22.1371 | 3815.604 |

**Table 9.** Minimum and average values of objective 1

| Maxiter | NSGA-II | | SPEA2 | |
|---|---|---|---|---|
| | Minimum | Average | Minimum | Average |
| 10 | 18.29412 | 20.46108 | 17.8851 | 21.6870 |
| 20 | 15.80214 | 18.99706 | 16.1214 | 20.3548 |
| 30 | 14.51872 | 18.07329 | 16.013 | 21.5642 |
| 40 | 14.07487 | 16.67112 | 15.8705 | 20.6570 |
| 50 | 13.58289 | 16.14251 | 16.4506 | 18.1254 |
| 60 | 14.99465 | 17.50401 | 15.4512 | 19.5742 |
| 70 | 9 | 16.25561 | 15.3214 | 18.8542 |
| 80 | 13.11765 | 16.03107 | 14.1274 | 18.350 |
| 90 | 16.21925 | 17.30802 | 16.8750 | 17.0125 |
| 100 | 13.81283 | 15.04813 | 17.1234 | 17.5687 |

**Table 10.** Minimum and average values of objective 2

| Maxiter | NSGA-II | | SPEA2 | |
|---|---|---|---|---|
| | Minimum | Average | Minimum | Average |
| 10 | 4243.933 | 7151.112 | 3975.664 | 7542.325 |
| 20 | 2565.571 | 5679.284 | 3426.157 | 7124.56 |
| 30 | 2937.743 | 5307.293 | 3687.1235 | 6875.2217 |
| 40 | 2547.914 | 3510.838 | 2857.451 | 4150.238 |
| 50 | 2744.762 | 3373.973 | 2513.5469 | 4215.3256 |
| 60 | 2341.2 | 2659.411 | 2587.325 | 3548.786 |
| 70 | 2547.914 | 3510.838 | 3456.217 | 4687.125 |
| 80 | 2357.874 | 3291.238 | 2678.650 | 3015.864 |
| 90 | 2307.8 | 2993.839 | 2546.254 | ٣١٥٤,٦٥ |
| 100 | 2018.9 | 3291.586 | 2840.25 | 3345.481 |

## 6. Conclusions

In this research, a mathematical model is designed for re-entrant flow shop scheduling problem. This system has $m$ stations which can process several products in a moment. Two objectives have been considered for this problem. One of them is the maximum rate of production and the other is the minimization of process time. Because this problem is accounted as NP-hard problem, NSGA-II algorithm is employed to solve this problem. To set algorithm's parameters, Taguchi experiment designing methods is employed and efficiency and effectiveness of algorithm have been shown using several numerical examples. To validate the proposed algorithm, results are compared with SPEA2 and the results show that the NSGA-II performs better than the SPEA2. To conduct future researches, the following cases can be mentioned: Considering characteristics of recursive production in the job-shop production systems which increases problem complexity, using other meta-heuristic methods such as multi-objective particle swarm optimization algorithm (MOPSO) and multi-objective Imperialist Competitive algorithm and comparing their results with results obtained from algorithm recommended in this research.

## References

Belkaid, F.,  Yalaoui, F., &  Sari, Z. (2016). An Efficient Approach for the Re-entrant Parallel Machines Scheduling Problem under Consumable Resources Constraints. *International journal of Information Systems and Supply Chain Management*, 9 (3), July, 1-25.

Chen, J-S., Pan, J.C-H., & Wu, C-K. (2008). Hybrid tabu search for re-entrant permutation flow-shop scheduling problem. *Expert Systems with Applications*, 34, 1924–1930.

Choi, S-W., & Kim, Y-D. (2008). Minimizing makespan on an m-machine re-entrant flowshop. *Computers & Operations Research*, 35, 1684 – 1696.

Choi, S-W., & Kim, Y-D. (2009). Minimizing total tardiness on a two-machine re-entrant flowshop. *European Journal of Operational Research*, 199, 375–384.

Choi, J-Y., & KO, S-S. (2009). Simulation-based two-phase genetic algorithm for the capacitated re-entrant line scheduling problem. *Computers & Industrial Engineering*, 57, 660–666.

Dong, M., He, F. (2012). A new continuous model for multiple re-entrant manufacturing systems. *European Journal of Operational Research*, 223, 659–668.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-Ⅱ. *IEEE Trans*. Evol. Comput, 6 (2), 182–197.

Fattahi, P., Tavakoli, N.B., Jalilvand-Nejad, A., & Jolai, F. (2010). A hybrid algorithm to solve the problem of re-entrant manufacturing system scheduling. *CIRP Journal of manufacturing Science and Technology*, 3, 268-278.

Cavory, G., Dupas, R., & Goncalves, G. (2005). A Genetic Approach to Solving the Problem of Cyclic Job Shop Scheduling with Linear Constraints. *European Journal of Operational Research*, 161, 73–85.

Hinze, R. (2015). A Lot Streaming Model for a Re-entrant Flow Shop Scheduling Problem with Missing Operations. *Logistics Management*, 149-158.

Hsu, T., Korbaa, O., Dupas, D., & Goncalves, G. (2008). Cyclic Scheduling for F.M.S.: Modeling and Evolutionary Solving Approach. *European Journal of Operational Research*, 191, 464–484.

Hwang, H., & Sun, J.U. (1998). Production sequencing problem with re-entrant work flows and sequence dependent setup times. Int. J. Prod. Res., 36, 2435–2450.

Jeong, B., & Kim, Y-D. (2014). Minimizing total tardiness in a two-machine re-entrant flowshop with sequence-dependent setup times. Accepted Manuscript to appear in: *Computers & Operations Research.*

Jing, C., Tang, G., & Qian, X. (2008). Heuristic algorithms for two machine re-entrant flow shop. *Theoretical Computer Science*, 400, 137–143.

Jing, C., Huang, W., & Tang, G. (2011). Minimizing total completion time for re-entrant flow shop scheduling problems. *Theoretical Computer Science*, 412, 6712-6719.

Kia, H., Ghodsypour, S.H., & Davoudpour, H. (2017). New scheduling rules for a dynamic flexible flow line problem with sequence-dependent setup times. *Journal of Industrial Engineering International*, 13(1), 1–10.

Kim, S., Park, Y., & Jun, C-H. (2006). Performance evaluation of re-entrant manufacturing system with production loss using mean value analysis. *Computers & Operations Research*, 33, 1308–1325.

Kubale, M., & Nadolski, A. (2005). Chromatic Scheduling in a Cyclic Open Shop. *European Journal of Operational Research*, 164, 585–591.

Kumar, V.M., Murthy, A., & Chandrashekara, K. (2012). A hybrid algorithm optimization approach for machine loading problem in flexible manufacturing system. *Journal of Industrial Engineering International*, 8(1), 3-15.

Lee, C.K.M., & Lin, D. (2010). Hybrid genetic algorithm for bi-objective flow shop scheduling problems with re-entrant jobs. *IEEE International Conference on Industrial Engineering and Engineering Management*, IEEE Computer Society, Macao, China, 1240–1245.

Lin D., Lee, C.K.M., & Ho, W. (2013). Multi-level genetic algorithm for the resource-constrained re-entrant scheduling problem in the flow shop. *Engineering Applications of Artificial Intelligence*, 26, 1282-1290.

Mirabi, M., Fatemi Ghomi, S.M.T., & Jolai, F. (2014). A novel hybrid genetic algorithm to solve the make-to-order sequence-dependent flow-shop scheduling problem. *Journal of Industrial Engineering International*, 10(2), 57-69.

Rau, H., & Cho, K.H. (2009). Genetic algorithm modeling for the inspection allocation in re-entrant production systems. Expert Syst. Appl., 36, 11287–11295.

Teruo, M. (1990). The New Experimental Design, Taguchi's Approach to Quality Engineering. ASI Press, First Editon, Printed In The United States Of American.

Xu, J., Lin W-C., Wu, J., Cheng, S-R., & Wu, C-C. (2016).Heuristic based genetic algorithms for the re-entrant total completion time flowshop scheduling with learning consideration. *International Journal of Computational Intelligence Systems*, 9, 1082-1100.

Xu, J., Yin, Y., Cheng, T.C.E., Wu, C-C., & Gu, S. (2014). A memetic algorithm for the re-entrant permutation flowshop scheduling problem to minimize the makespan. *Applied Soft Computing*, 24, 277-283.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. Technical Report 103, *Computer Engineering and Networks Laboratory (TIK)*, ETH Zurich, Zurich, Switzerland.