Mazandaran University of Science and Technology

Available online at www.iiec2017.com

13th International Conference on Industrial Engineering (IIEC 2017)

Iran Institute of Industrial Engineering

Two models and an innovative genetic algorithm for scheduling in flexible manufacturing systems

Mehrdad Nouri Koupaei[a], Mohammad Mohammadi[b] and Bahman Naderi[c]

[a] *Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*
Phone: +98 (21) 88830891, Fax: +98 (21) 88329213, e-mail: mhrdd_nouri@yahoo.com

[b] *Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*
Phone: +98 (21) 88830891, Fax: +98 (21) 88329213, e-mail: Mohammadi@khu.ac.ir

[c] *Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*
Phone: +98 (21) 88830891, Fax: +98 (21) 88329213, e-mail: bahman.naderi@aut.ac.ir

Abstract

In recent decades, flexible manufacturing systems have emerged as a response to market demands of high product diversity. Scheduling is one important phase in production planning in all manufacturing systems. Although scheduling in classical manufacturing systems, such as flow and job shops, are well studied. Rarely, any paper studies scheduling of the more recent flexible manufacturing system. This paper investigates scheduling in the flexible manufacturing systems where there are both machine and routing flexibilities. In the first step, two mathematical models in form of mixed integer linear programs are proposed for the problem. The first model is position-based and the second is sequence-based. The models can solve optimally small problems. In the second step, since the problem is NP-hard, we develop an efficient genetic algorithm for large scale problems, using the properties of the optimal schedule. Finally, we carry out computational experiments to demonstrate the effectiveness of our algorithm. The results show that the proposed algorithm has the ability to achieve the good solutions in reasonable computational time.

Keywords:
scheduling, flexible manufacturing systems, mathematical model, genetic algorithm

Introduction

Nowadays, due to competitive market conditions, organizations need to increase their production efficiency and optimize operations. Therefore, in order to optimize their schedule and to online respond, organizations must consider different factors to the demands of customers. With integrated planning systems, organizations can quickly respond to changes and produce higher quality products and lower production costs. Each process can be optimized and complex problems modeled in engineering, economic and commercial for optimization problems. Today, FMSs seem to be a very promising technology as they provide flexibility, which is essential for many manufacturing companies to stay competitive in a highly dynamic and changing manufacturing environment. The optimal scheduling of FMS is a critical issue and it is a complex problem. Flexible manufacturing systems (FMSs) have been developed to combine the flexibility of job shops and the productivity of flow lines. Scheduling in FMSs differs from that in a conventional job shop because of the availability of alternating manufacturing resources resulting in routing flexibility. This may potentially increase the output by eliminating the bottle-necks often present when alternate routes are not feasible. FMS is a highly integrated manufacturing system and the inter-relationships between its various components are not well understood for a very complex system. Some of optimization problems are complex and obtain optimal solutions in a reasonable time with the exact solution such as dynamic programming and branch and bound methods is difficult. So the development of these types of problems that can be solved in reasonable time optimal or acquire near-optimal solutions are suitable economically. In recent years, researchers in optimizing the most complex problems by implementation of meta heuristic methods have achieved good results. Due to this complexity, it is difficult to accurately calculate the performance measures of the FMS which leads to its design through mathematical techniques. Various combination of objectives can be considered for scheduling problem of an FMS. Flexible manufacturing systems have been developed to combine the flexibility of open shops and the productivity of job shops. A flexible manufacturing system (FMS) are includes a set of n jobs and a set of m machine. Each job consists of a chain of operations, each of which needs to be processed during an immediately on a given machine. Each machine can process at most one operation

at a time.
This paper focuses on a new scheduling approach based on genetic can be used to complement the currently used computer-based scheduling systems. The proposed algorithm is based on two assumptions: first, each machine is eligible to process operation. Second, the number of operations that assigned to each machine is eligible to process. For modeling this problem, two mathematical models have been proposed: The first model is position-based. The second model is sequence-based.

Literature review

In the literature, different algorithms such as Simulated Annealing, Artificial Immune System, Particle Swarm Optimization (PSO), Petri Net, Tabu Search, Branch and Bound and integer programming have been used to solve the FMS scheduling problem. Researchers always have been intrigued by the scheduling problem in FMS system because of its importance in today's manufacturing industry. Over the past 10 to 15 years, most research on the scheduling of FMSs has focused on developing scheduling algorithms for a single objective such as minimization of total tardiness and maximization of throughput, and so on. The present work, however, considers multi-objective functions in the development of the scheduler to utilize resources maximally, thereby offsetting the high installation cost of equipment. Thus the multiple objectives must be achieved to satisfy both customers and the FMS company.
The first mathematical formulation for FMS-loading problem was given by Stecke [1]. He formulated the production planning problem as non-linear 0–1 mixed integer programs and applied several linear methods on the data from an existing FMS. Udhayakumar, and Kumanan presented an integrated scheduling of flexible manufacturing system with different priority dispatching rules using evolutionary algorithms [2]. Jerald et al. solved a scheduling optimization of flexible manufacturing systems using particle swarm optimization algorithm. The FMS scheduling problem has been tackled by various traditional optimization techniques. While these methods can give an optimal solution to small-scale problems, they are often inefficient when applied to larger-scale problems. In this work, different scheduling mechanisms are designed to generate optimum scheduling; these include non-traditional approaches such as genetic algorithm (GA), simulated annealing (SA) algorithm, memetic algorithm (MA) and particle swarm algorithm (PSA) by considering multiple objectives, i.e., minimizing the idle time of the machine and minimizing the total penalty cost for not meeting the deadline concurrently [3].
Reddy and Rao presented a hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS [4]. Huang and Liao presented an ant colony optimization combined with taboo search for the job shop scheduling problem [5]. In other research, Fauadi and Murata solved makespan minimization of machines and automated guided vehicles schedule using binary particle swarm optimization [6]. Baruw and Piera [7]. Some research using a genetic algorithm to solve the FMS scheduling problem. Zakaria and Petrovic [8] proposed a genetic algorithm for match-up rescheduling with non-reshuffle and reshuffle strategies which accommodate new orders by manipulating the available idle times on machines and by sequencing operations, respectively. Kumar et al. [9] presented a new integrated approach to concurrently address the machine loading and the tool allocation problems in an FMS environment. Candan and Yazgan [10] Studied on Genetic algorithm parameter optimization using Taguchi method for a flexible manufacturing system scheduling problem. Also, the another research using genetic algorithm was studied by Wu et al. [11], Soolaki and Zarrinpoor [12], Abazari et al. [13], Filho et al. [14] and Umar et al. [15].
Priore et al. presented a comparison of machine-learning algorithms for dynamic scheduling of flexible manufacturing systems [16]. Noorul Haq et al. [17] discussed the multilevel scheduling decisions of a FMS to generate realistic schedules for the efficient operation of the FMS. Abdelmaguid et al. [18] has presented a new hybrid genetic algorithm for the simultaneous scheduling problem for the makespan minimization objective.

Problem formulation

In this paper, the flexible manufacturing systems (FMS) have been investigated by considering the following assumptions: first, each machine is eligible to process operation. Second, the number of operations that assigned to each machine is eligible to process.
Two mathematical models have been proposed for this problem. The first model is position-based. The second model is sequence-based. This problem has the ability to achieve the optimum for small problems. These problems are first formulated as a mixed integer linear programming model. Using these models, small instances are solved for optimality.

Parameters and indices:

n	The number of jobs
m	The number of machine
j, h	Index for jobs $\{1, 2, \dots, n\}$
l, b	operations of job i $\{1, 2, \dots, t_j\}$
i	Index for machines $\{1, 2, \dots, m\}$
k	positions in machine i
t_j	The number of operations of job *j*.
$e_{i,l}$	A parameter that takes value 1 if machine *i* is eligible to process operation *l*, and 0 otherwise.
$p_{j,l,i}$	The processing time of operation *l* of job *j* on machine i.
v_i	The number of operations that machine *i* is eligible to process.
M	A large positive number

Model 1 (position-based)
In this model, binary variables specify the operations' position in both processing route of a job and job order of a machine; so, it is called position-based model. The

following notations are established:

Variables:

$X_{j,l,i,k}$ Binary variable taking value 1 if $O_{j,l}$ is processed in position k of machine i, and 0 otherwise.

$Y_{j,l,b}$ Binary variable taking value 1 if $O_{j,l}$ is processed immediately after $O_{j,b}$, and 0 otherwise.

$C_{j,l}$ Continuous variable for the completion time of $O_{j,l}$.

Min C_{max}

Subject to:

$$\sum_{i=1}^{m}\sum_{k=1}^{v_i} X_{j,l,i,k} = 1$$
$$\sum_{j=1}^{n}\sum_{l=1}^{t_j} X_{j,l,i,k} \le 1$$
$$\sum_{j=1}^{n}\sum_{k=1}^{v_i} X_{j,l,i,k} \le n \cdot e_{i,l}$$
$$\sum_{b=0,b\neq l}^{t_j} Y_{j,l,b} = 1$$
$$\sum_{l=1,l\neq b}^{t_j} Y_{j,l,b} \le 1$$
$$\sum_{l=1}^{t_j} Y_{j,l,0} = 1$$
$$C_{j,l} \ge C_{j,b} + \sum_{i=1}^{m}\sum_{k=1}^{v_i} X_{j,l,i,k} \cdot p_{j,l,i} - M \cdot \left(1 - Y_{j,l,b}\right)$$
$$C_{j,l} \ge C_{h,b} + p_{j,l,i} - M \cdot \left(2 - X_{j,l,i,k} - \sum_{f=1}^{k-1} X_{h,b,i,f}\right)$$
$$C_{max} \ge C_{j,l}$$
$$C_{j,l} \ge 0$$
$$X_{j,l,i,k}, Y_{j,l,b} \in \{0,1\}$$

where the $C_{j,0} = 0$.

Constraint set (1) states that every operation in each positions of a machine is scheduled once. Constraint set (2) ensure that every operation must have at most one succeeding operation in processing route of the jobs and in job order of every machine and positions. Constraint sets (3) enforce that a job can be assigned to a machine in each positions that this machine is eligible to process operation. Constraint set (4) and (5) states that all operations processed must done immediately after another operation. Constraint set (6) enforce that dummy job 0 and operation 0 must have exactly one successor. Constraint set (7) ensures that O_{jl} cannot start before O_{jb} completes if O_{jb} precedes O_{jl}. Constraint set (8) assures that O_{jl} begins only after O_{hb} finishes if O_{jl} occupies a position after O_{hb}. Constraint set (9) computes makespan. Constraint (10) indicates that continuous variable for the completion time of O_{jl} should be available for scheduling at time zero. Constraint sets (11) define the decision variables.

Model 2 (Sequence-based)

Model 2 uses binary variables that show the relative sequence of different operations of a job as well as the relative order of the jobs on each machine. The following notations are established:

Variables:

$X_{j,l,h,b}$ Binary variable taking value 1 if Oj,l is processed after Oh,b, and 0 otherwise. j < n, h > j

$Y_{j,l,b}$ Binary variable taking value 1 if Oj,l is processed after Oj,b, and 0 otherwise. l < m, b > l

$Z_{j,l,i}$ Binary variable taking value 1 if Oj,l is processed on machine i.

$C_{j,l}$ Continuous variable for the completion time of Oj,l.

Min C_{max}

Subject to:

$$\sum_{i=1}^{m} Z_{j,l,i} = 1 \qquad \forall_{j,l}$$
$$Z_{j,l,i} \le e_{i,l} \qquad \forall_{i,k}$$
$$C_{j,l} \ge C_{j,b} + \sum_{i=1}^{m} Z_{j,l,i} \cdot p_{j,l,i} - M \cdot \left(1 - Y_{j,l,b}\right) \qquad \forall_{i,l}$$
$$C_{j,b} \ge C_{j,l} + \sum_{i=1}^{m} Z_{j,b,i} \cdot p_{j,b,i} - M \cdot Y_{j,l,b} \qquad \forall_{j,l}$$
$$C_{j,l} \ge C_{h,b} + p_{j,l,i} - M \cdot \left(3 - X_{i,j,h,b} - Z_{j,l,i} - Z_{h,b,i}\right) \qquad \forall_{j,b}$$
$$C_{h,b} \ge C_{j,l} + p_{h,b,i} - M \cdot X_{i,j,h,b} - M \cdot \left(2 - Z_{j,l,i} - Z_{h,b,i}\right) \qquad \forall_{j}$$
$$C_{max} \ge C_{j,l} \qquad \forall_{j,l,}$$
$$C_{j,l} \ge 0$$
$$X_{j,l,h,b}, Y_{j,l,b}, Z_{j,l,i} \in \{0,1\} \qquad \forall_{j,l,}$$

Constraint set (1) states that every machine must perform an operation in each time. Constraint sets (2) enforce that a machine can only perform the operations that have eligible to process this operation. Constraint set (3) ensures that O_{jl} cannot start before O_{jb} completes if O_{jb} precedes O_{jl}. Also, Constraint set (4) ensures that O_{jb} cannot start before O_{jl} completes if O_{jl} precedes O_{jb}. Constraint set (5) assures that O_{jl} begins only after O_{hb} finishes if O_{jl} occupies a position after O_{hb}. Also, Constraint set (6) assures that O_{hb} begins only after O_{jl} finishes if O_{hb} occupies a position after O_{jl}. Constraint set (7) computes makespan. Constraint (8) indicates that continuous variable for the completion time of O_{jl} should be available for scheduling at time zero. Constraint sets (9) define the decision variables.

Designing the proposed algorithm (solution method)

The GA was developed by John Holland and students in the 1970s. Apart from traditional optimization methods, a code format instead of a parameter set is employed in the GA [10]. The GA works according to the rules of probability and requires only an objective function. The algorithm searches a specific part of the solution rather than an entire solution [15, 26].

Thus, a solution is expected to be attained in a much shorter time [27]. The GA can be applied to attaining the solution of large problems which have a large number of factors affecting a solution space. At any given iteration, the genetic algorithm operates on a pool of solutions rather than a single solution. In GA, search starts with an initial set of random solutions known as population [9, 28, 29]. In this

study, a metaheuristic genetic algorithm (GA) technique commonly used as a stochastic search method in recent years was employed and an appropriate schedule was obtained to assigning operations to machines and sequencing operations assigned to each machine. The objective of the problem was to minimize the makespan (Cmax) of an FMS scheduling problem.

Chromosome representation

In general, the process of encoding solutions includes two steps: In the first phase, operations allocated to machines. Then, in the second phase, the operation sequencing is determined. For this reason, two-dimensional chromosome is defined that the first dimensional are includes the machines vector and the second dimensional are includes the operation sequencing vector.
For example, there are four jobs that two operations must be done on each jobs. There are two machines to perform each operation. So one of the possible solutions to the following:
a: [2 2 2 1 1 1 2 1]
h: [7 3 5 4 6 1 8 2]
This solution show that the seventh operations are done on the second machine. Next, the third operations are done on the second machine. Next, the fifth operations are done on the second machine. Then, the fourth operations are done on the first machine. Next, the Sixth operations are done on the first machine. Next, the first operations are done on the first machine. Then, the eighth operations are done on the second machine. Finally, the second operations are done on the first machine.

Initial population

This process starts with initializing a population. It means that a population of initial solutions is randomly generated over the problem space. The fitness of the weeds initialized is evaluated depending upon the fitness function or the objective function chosen for the optimization problem.
This section consists of 2 stages:

1. allocating operations to machines: At this stage, randomly selecting an operation from virtual operations for the processing.
2. operation sequencing: To create a feasible initial solution to the problem, first, all operations have been put on the list of unplanned operations. Also, for each of them, a unique random number in the interval [0, 1] is generated.

A population of initial solutions is randomly generated over the problem space. The fitness of the weeds initialized is evaluated depending upon the fitness function or the objective function chosen for the optimization problem.

Selection and reproduction

The selection creates an opportunity to deliver the gene of a good obtained solution to the next generation. In this paper, the roulette wheel selection is utilized where chromosome selection in mating pool is based on their probability selection. The probability selection of each chromosome is evaluated based on its fitness value.

Crossover operator

Crossover exchanges some of genes of the chromosomes through the breakage and reunion of two selected chromosomes in order to generate a number of children. A predetermined parameter pCrossover=0.8 is used to represent the probability of crossover operator applied in a population. The arithmetic crossover is utilized for this purpose. The formula for calculating the number of operator is as follows:
nCrossover=round(pCrossover*nPop/2)*2

Mutation operator

Mutation operator makes an offspring solution by randomly modifying the parent's features. This operator helps to generate a reasonable level of diversity in the population. It also serves the search by jumping out of local optimal solutions. In this research, an exchange mutation is chosen. This mutation swaps value of the two random selected genes of current solution together. A predetermined parameter pMutation=0.3 is used to represent the probability of mutation operator applied in a population. This operator is implemented in each section of the chromosome. The search process of the algorithm stops if the number of generations is greater than a maximum number of generations or some specified number of generations without improvement of best-known solution is reached. The formula for calculating the number of operator is as follows:
nMutation=round(pMutation*nPop)

Evaluation and validation of results

Because the innovation of this research is in both the model and the solution method, Therefore, the computational data analysis to determine the validity of the model and determine the efficiency of solution will be undertaken. In section (5.1), the validity of the proposed model using numerical examples in compact size and compare answers with the exact solution is shown.
In section (5.2), to demonstrate the ability of the algorithm in convergence to the optimal solution, we use the exact solution in numerical examples with small size. Also, to demonstrate the speed and quality of the algorithm, we use the numerical examples with medium and large size.
We used GAMS 22.2 to calculate the exact solution of the problem. In addition, to implement the proposed genetic algorithm MATLAB R 2011is used.

Validation of the proposed model

In this section, the validity of the proposed model using numerical examples in compact size and compare answers with the exact solution is shown.

Consider four job that each job must be performed with two operations. Two machines can do these operations. Each job can be performed by one machine at a time. Each machine at a time can perform only one operation.
In the table 1, the process time of each operation on each machines for all jobs shown.

Table 1. Numerical example problem in small size

Jobs	Machine 1		Machine 2	
	Operation 1	Operation 2	Operation 1	Operation 2
Job 1	2	3	4	5
Job 2	-	1	4	-
Job 3	2	6	3	-
Job 4	-	1	4	6

The goal is to find the best production scheduling so that each operation of jobs is done and the time of all operations minimized. Using commonality counting, the optimal solution is: C_{max}=13 that this Gantt chart is shown below:

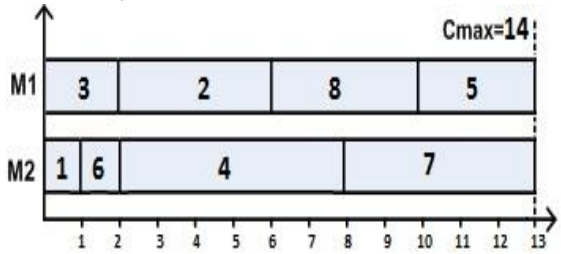

Fig 1. Optimal scheduling gnat chart of numerical example

Validation of the proposed algorithm and analysis the results

To evaluate the efficiency of the algorithm, 40 problems in large size, medium and small were produced. Given that genetic algorithms basically are random algorithms, therefore, this algorithm was run 10 times for each problem and the best solution obtained was considered as a final solution of problem.
In order to increase the quality of proposed algorithm solutions, the most suitable parameter values by numerical experiments obtained that shown in Table 2 below:

Table 2. The most appropriate algorithm parameters

parameters	**Value**		
	small size	**medium size**	**large size**
Crossover rate (nCrossover)	0.8	0.8	0.85
Mutation rate (nMutation)	0.3	0.3	0.25
Population size (nPop)	15	15	20
Number of generations (MaxIt)	200	300	400
Tournament Selection Size	3	3	3

Comparing the proposed algorithm with GAMS in terms of solutions quality

Obtained solutions in different dimensions are shown in table 3. The purpose of comparing two solutions is evaluating the ability of the proposed algorithm to achieve the optimal or near-optimal solutions.

Table 3. The information of GAMS and Proposed Algorithm in different dimensions

Sample	**Size**	**GA (Cmax)**	**GA (CPU Time)**	**GAMS (Cmax)**	**GAMS (CPU Time)**
1	**m=5, m=5, O=5**	18	5.2031	18	23.2
2	**m=5, m=5, O=5**	15	5.1719	15	21.8
3	**m=5, m=5, O=5**	13	5.1875	13	24.3
4	**m=5, m=5, O=5**	18	5.0313	16	25.4
5	**m=5, m=5, O=5**	17	5	16	26.1
6	**m=10, m=5, O=5**	26	10.4219	-	-
7	**m=10, m=5, O=5**	29	10.0469	-	-
8	**m=10, m=5, O=5**	19	10.0938	-	-
9	**m=10, m=5, O=5**	30	10.4688	-	-
10	**m=10, m=5, O=5**	30	10.6563	-	-
11	**m=15, m=5, O=5**	39	15.1094	-	-
12	**m=15, m=5, O=5**	41	15.7344	-	-
13	**m=15, m=5, O=5**	30	15.9531	-	-
14	**m=15, m=5, O=5**	33	15.2344	-	-
15	**m=15, m=5, O=5**	33	16.5313	-	-
16	**m=20, m=5, O=5**	44	21.0625	-	-
17	**m=20, m=5, O=5**	45	21.3125	-	-

18	**m=20, m=5, O=5**	53	20.8281	-	-
19	**m=20, m=5, O=5**	48	21.1875	-	-
20	**m=20, m=5, O=5**	45	20.4063	-	-
21	**m=5, m=10, O=10**	20	5.5313	-	-
22	**m=5, m=10, O=10**	25	5.2031	-	-
23	**m=5, m=10, O=10**	24	5.2813	-	-
24	**m=5, m=10, O=10**	24	5.4219	-	-
25	**m=5, m=10, O=10**	23	5	-	-
26	**m=10, m=10, O=10**	20	5.1719	-	-
27	**m=10, m=10, O=10**	23	5.2656	-	-
28	**m=10, m=10, O=10**	23	5.6719	-	-
29	**m=10, m=10, O=10**	22	5.3594	-	-
30	**m=10, m=10, O=10**	22	5.5938	-	-
31	**m=15, m=10, O=10**	33	15.8125	-	-
32	**m=15, m=10, O=10**	32	16.9375	-	-
33	**m=15, m=10, O=10**	28	16.7188	-	-
34	**m=15, m=10, O=10**	30	16.3125	-	-
35	**m=15, m=10, O=10**	33	17	-	-
36	**m=20, m=10, O=10**	34	20.9688	-	-
37	**m=20, m=10, O=10**	34	23.9063	-	-
38	**m=20, m=10, O=10**	35	20.4531	-	-
39	**m=20, m=10, O=10**	37	20.2813	-	-
40	**m=20, m=10, O=10**	36	21.0469	-	-

Conclusion and future work

This paper, proposes a new methodology, constraint-based genetic algorithm (CBGA) to handle a complex variety of variables and constraints in a typical FMS-loading problem. The primary objective in this research was to develop a methodology that minimizes the manufacturing makespan within a FMS environment, while reducing the time that is required to develop and produce a realistic production schedule. To achieve this aim, three new genetic operators—constraint based: initialization, crossover, and mutation are introduced. The methodology developed here helps avoid getting trapped at local minima. In this paper, the flexible manufacturing systems (FMS) have been investigated by considering the following assumptions: first, each machine is eligible to process operation. Second, the number of operations that assigned to each machine is eligible to process. Two mathematical models have been proposed for this problem. The first model is position-based. The second model is sequence-based. This problem has the ability to achieve the optimum for small problems. These problems are first formulated as a mixed integer linear programming model. Using these models, small instances are solved for optimality.

The application of the algorithm is tested on standard data sets and its superiority is demonstrated. The solution approach is illustrated by a simple example and the robustness of the algorithm is tested on 40 problems in large size, medium and small were produced. Given that genetic algorithms basically are random algorithms, therefore, this algorithm was run 10 times for each problem and the best solution obtained was considered as a final solution of problem. Finally, the most effective factors and their levels were employed in the proposed GA to compare against the problem from the literature. The results showed that the proposed GA algorithm produces better result than the others.

References

[1] Stecke KE, 1983, Formulation and solution of non-linear integer production planning problem for flexible manufacturing system. Manag Sci 29:273–288.

[2] Udhayakumar, P. & Kumanan, S., 2012, Integrated scheduling of flexible manufacturing system using evolutionary algorithms, The International Journal of Advanced Manufacturing Technology, Volume 61, Issue 5, pp 621–635.

[3] Jerald, J., Asokan, P., Prabaharan, G., 2005, Scheduling optimisation of flexible manufacturing systems using particle swarm optimisation algorithm, The International Journal of Advanced Manufacturing Technology, Volume 25, Issue 9, pp 964–971.

[4] Reddy BSP, Rao CSP, 2006, A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS The International Journal of Advanced

Manufacturing Technology, Volume 31, Issue 5–6, pp. 602–613.
[5] Huang KL, Liao CJ, 2008, Ant colony optimization combined with taboo search for the job shop scheduling problem, Computer Operation Research, Volume 35, Issue4, pp. 1030–1046.
[6] Fauadi MHF, Murata T, 2010, Makespan minimization of machines and automated guided vehicles schedule using binary particle swarm optimization, Proceedings of the International Multi Conference of Engineers and Computer Scientists 2010, volume 3, IMECS 2010, Hong Kong.
[7] Baruw. O. T. and Pier. M. A., 2015, Identifying FMS repetitive patterns for efficient search-based scheduling algorithm: A colored Petri net approach, Journal of Manufacturing Systems 35 (2015), pp. 120–135.
[8] Zakaria and Petrovic, 2012, Genetic algorithms for match-up rescheduling of the flexible manufacturing systems, Computers & Industrial Engineering, Volume 62, Issue 2, pp. 670-686.
[9] Kumar. A., Prakash, Tiwari. P. M., Shankar. R., Bavej. A., 2006, Solving machine-loading problem of a flexible manufacturing system with constraint-based genetic algorithm, European Journal of Operational Research, Volume 175, Issue 2, pp. 1043–1069.
[10] Candan. G. and Yazgan. H. R., 2015, Genetic algorithm parameter optimisation using Taguchi method for a flexible manufacturing system scheduling problem, International Journal of Production Research, (2015), Vol. 53, No. 3, pp. 897–915.
[11] Wu. K.Y., Xu. S.S. and Wu. T.C., 2013, Optimal Scheduling for Retrieval Jobs in Double-Deep AS/RS by Evolutionary Algorithms, Abstract and Applied Analysis, Volume 2013 (2013), Article ID 634812, http://dx.doi.org/10.1155/2013/634812.
[12] Soolaki. M., Zarrinpoor. N., 2014, A new 0-1 linear programming approach and genetic algorithm for solving assignment problem in flexible manufacturing system, The International Journal of Advanced Manufacturing Technology, Volume 75, Issue 1, pp. 385-394.
[13] Abazari. A. M., Solimanpur. M. and Sattari. H., 2012, Optimum loading of machines in a flexible manufacturing system using a mixed-integer linear mathematical programming model and genetic algorithm, Computers & Industrial Engineering, Volume 62, Issue 2, pp. 469–478.
[14] Filho. M. G., Barco. C. F. and Neto. R. T., 2012, Using Genetic Algorithms to solve scheduling problems on flexible manufacturing systems (FMS): a literature survey, classification and analysis, Flexible Services and Manufacturing Journal, Volume 26, Issue 3, pp. 408-431.
[15] Umar. U. A., Ariffin. M. K., Ismail. N. and Tang. S. H., 2015, Hybrid multiobjective genetic algorithms for integrated dynamic scheduling and routing of jobs and automated-guided vehicle (AGV) in flexible manufacturing systems (FMS) environment, The International Journal of Advanced Manufacturing Technology, pp. 1-19.
[16] Priore, P., de la Fuente, D., Puente, J. and Parreño, J., 2006, A comparison of machine-learning algorithms for dynamic scheduling of flexible manufacturing systems, Engineering Applications of Artificial Intelligence, Volume 19, Issue 3, pp. 247-255.
[17] Noorul Haq A, Karthikeyan T, Dinesh M, 2003, Scheduling decisions in FMS using a heuristic approach. The International Journal of Advanced Manufacturing Technology, Volume 23, Issue 1, pp. 374–379.
[18] Abdelmaguid TF, Nassef AO, Kamal BA, Hassan MF, 2004, A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research, Volume 42, Issue 2, pp. 267–281.