American Journal of Applied Sciences 9 (10): 1706-1720, 2012
ISSN 1546-9239

Scheduling of Automated Guided Vehicle and Flexible Jobshop using Jumping Genes Genetic Algorithm

[1]Paul Pandian, P., [2]S. Saravana Sankar,
[3]S.G. Ponnambalam and [4]M. Victor Raj
[1]Department of Mechanical Engineering,
Sethu Institute of Technology Kariapatti-626 115, India
[2]Department of Mechanical Engineering Kalasalingam,
University Krishnankoil-626 190, India
[3]Department of Mechatronics,
Monash University, Petaling Jaya, Selangor, Malaysia
[4]Department of Mechanical Engineering,
Dr. Sivanthi Aditanar College of Engineering-628 215, India

Abstract: Problem statement: Now a day's many researchers try Genetic algorithm based optimization to find near optimal solution for flexible job shop. It is a global search. In Our study in the GA, some changes are made to search locally and globally by adding jumping genes operation. A typical flexible job shop model is considered for this research study. For that layout, five different example problems are formulated for purpose of evaluation. The material flow time for different shop types, processing times of products, waiting times of products, sequences of products are created and given in tabular form. **Approach:** The one of best evolutionary approach i.e., genetic algorithm with jumping genes operation is applied in this study, to optimize AGV flow time and the performance measures of Flexible Job shop manufacturing system. The non dominated sorting approach is used. Genetic algorithm with jumping genes operator is used to evaluate the method. **Results:** The AGV flow sequence is found out. Using this flow sequence make span, flow time of products with AGV, completion of the products is minimized. The position of the shop types are calculated for all products. The effectiveness of the proposed method is proved by comparing with Hamed Fazlollahtabar method. **Conclusion:** It is found that jumping genes genetic algorithm delivered good solutions as like as other evolutionary algorithms. Jumping genes genetic algorithm may applied to Multi objective optimization techniques in future.

Key words: Flexible jobshop manufacturing system, automated guided vehicle, jumping genes GA

INTRODUCTION

In the classical Job-Shop Scheduling Problem (JSP), n jobs are processed to completion on m unrelated machines. In order to match today's market requirements, manufacturing systems have to become more flexible (Saidi-Mehrabad and Fattahi, 2007). In the modern manufacturing plant, a machine has the capability of processing more than one type of operation. This leads to a modified version of JSP called the flexible JSP (Chen *et al*., 2008).

Flexible Job-Shop Scheduling Problem (FJSP) is an extended traditional job-shop scheduling problem. It breaks the restriction of unique resources and allows each operation to be processed by several different machines, thus making the job-shop scheduling problem accord with actual production situation more closely. FJSP is more complicated than the Job-Shop Scheduling Problem (JSP), since it needs to assign each operation to a machine from a set of capable machines and then sequence the assigned operations on each machine, referring to the study by (Xia and Wu, 2005).

Brucker and Schlie (1990) initially proposed the problem that one operation could be processed on several machines and have studied this problem deeply as pioneer. This marks the beginning of the study on FJSP. The methods for solving this kind of problem can be concluded into hierarchical approaches and integrated approaches. Hierarchical approach, which was firstly proposed by Brandimarte (1993), considered

Corresponding Author: Paul Pandian, P., Department of Mechanical Engineering, Sethu Institute of Technology Kariapatti-626 115, India

the assigning sub-problem and the sequencing sub-problem separately. Its basic idea is decomposing the complex problem into some sub-problems in order to decrease the complexity.

However, the integrated approaches solve the assigning sub-problem and the sequencing sub-problem simultaneously, such as greedy heuristics (Kannaiah *et al*., 2011), Simulated Annealing (SA) algorithm (Yussof *et al*., 2011) and Tabu Search (TS).

Most of the research on FJSP has been concentrated on single objective. However, several objectives must be considered simultaneously in the real-world production situation.

Recently, multi-objective FJSP has gained attention of some researchers. Kacem *et al*. (2002a; 2002b) used an approach by localization and multi-objective evolutionary optimization and proposed a Pareto approach based on the hybridization of Fuzzy Logic (FL) and Evolutionary Algorithms (EAs) to solve the FJSP. Xia and Wu (2005) proposed a practical hierarchical solution approach for solving MOFJSP. The proposed approach utilizes Particle Swarm Optimization (PSO) to assign operations on machines and Simulated Annealing (SA) algorithm to schedule operations on each machine. Liu *et al*. (2006) proposed the Variable Neighborhood Particle Swarm Optimization (VNPSO) consisting of a combination of the Variable Neighborhood Search (VNS) and Particle Swarm Optimization (PSO) for solving the multi-objective flexible job-shop scheduling problems. Ho and Tay (2007) presented an efficient approach for solving the multi-objective flexible job-shop by combining evolutionary algorithm and guided local search. They also solved the multi-objective flexible job-shop problems by using dispatching rules discovered through genetic programming (Tay and Ho, 2008). Gao *et al*. (2007) developed a hybrid Genetic Algorithm (GA) for the FJSP with three objectives: min makespan, min maximal machine workload and min total workload. Zhang *et al*. (2009) combined the PSO algorithm and Tabu Search (TS) algorithm for the multi-objective flexible job-shop problem. Xing *et al*. (2009) proposed an efficient search method for the multi-objective flexible job-shop scheduling problems.

Unordered subsequence Exchange crossover is tried by Thmilselvan and Balasubramanie (2012) in their work. Ripon *et al*. (2006) describes the jumping gene GA evolutionary algorithm that it imitates the jumping genes phenomenon discovered by Barbara McMlintock in which the induction of transposition of genes, within the chromosome itself that it consists or others. The concept of jumping genes is to provide local search capability to fine tune the scheduling solutions during evolution and produce a set of well converged and diverged solutions for job shop problem. Mansour (2011) developed a mathematical nonlinear integer programming model with stochastic controllable processing.

Although some improvements regarding optimization in FJMS have been achieved, heuristic algorithms to solve multi-objective AGV based FJMS. Fazlollahtabar *et al*. (2010) developed a mathematic programming approach to optimize material flow in an AGV-based FJMS. In this study, jumping genes GA is proposed to optimize material flow and makespan in an AGV-based jobshop manufacturing system Ripon *et al*. (2006) solve using jumping genes GA for jobshop problem.

Problem descriptions and assumptions: Here, we consider a jobshop layout which employs an AGV for material handling. The AGV carries raw material, semi produced and final products in batch sizes. Because of the increase in demands, advance in technology and rise in the production capacity, more shops than the existing shops are required. The new shops are associated with higher-technology machines. Therefore, more than one shop with the same performance is evolved. The difference among shops with the same performance is machines with various specifications that effect the production time/cost and productivity. As a result, the system could be a flexible jobshop model where multishops of the same performance exist and each operation is possible to be processed on any type of machine. The sequences of jobs are specified and the jobs are independent. To evaluate the performance of the proposed manufacturing system, we assess the material flow between any two shops of different types. In the proposed model, the aim is to optimize the material flow, i.e., finding a set of shops which minimize the material flow throughout the system and makespan. Here, flow is considered as the distance which the AGV moves to satisfy the production plan and demand. The proposed model is presented schematically in Fig. 1.

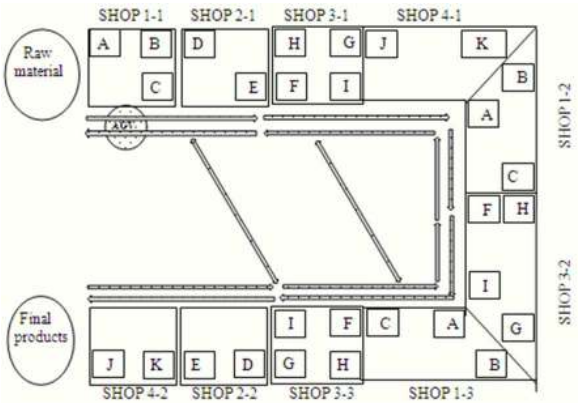

Fig. 1: Layout of flexible Jopshop

The sequence of the jobs is {1, 3, 2, 4}. After processing jobs 1 and 3, the accumulated total processing time is 10+15 = 25. Now, it is impossible to process the third job (job 2) because it has the processing time of 20 min, which would result in an accumulated total processing time of 45 min.

MATERIALS AND METHODS

The proposed mathematical model of the flexible jobshop problem is represented below. The indices, parameters and decision variables are as follows:
Indices:

M = Index for shops, m=1, 2,…, M
K = Index for shops, k=1, 2,…, K
N = Index for shop type m^{th}, n=1, 2,…, N
H = Index for shop type k^{th}, h=1, 2,…, H
I = Index for products, i=1, 2,…, I
P = Index for job position, p=1, 2,…, P

Parameters:
$Cipm_n$ = Completion time of product i^{th} in position p^{th} in shop m^{th} of type n^{th}
$P.Tim_n$ = Processing time of shop m^{th} of type n^{th} for product i^{th}
$T.Tim_nk_k$ = Transferring time from shop m^{th} of type n^{th} toshop k^{th} of type h^{th} for product i^{th}
V_{AGV} Velocity of AGV
fim_nk_k = Flow (distance) for product i^{th} between shop m^{th} of type n^{th} and shop k^{th} of type h^{th}
Wim_n = Waiting time for product i^{th} in shop m^{th} of type n^{th}
$C.T_i$ = Cycle time for product i^{th}
T = Total working time in each day:

$$Zipm = \begin{cases} 0, & \text{otherwise} \\ 1, & \text{if shop n th of type m th is chosen i th in position p th} \end{cases}$$

Decision variable:

$$Zipm = \begin{cases} 0, & \text{otherwise} \\ 1, & \text{if shop n th of type m th is chosen i th in position p th} \end{cases}$$

Objective function:

$$\text{Min } Z_l$$

$$\sum_{h=1}^{H}\sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{m=1;m=k}^{M}\sum_{p=1}^{P}\sum_{i=1}^{I} Zipm.Zi(p-1)kh.T\ Tim_nk_n \quad (1)$$

$$Cipm_n = \sum_{p-1}^{P}\sum_{n-1}^{N}\sum_{m-1}^{M}\left(z_{ipmn}(P.T_{imn} + W_{imn})\right) + \sum_{h-1}^{H}\sum_{k-1}^{k}\sum_{n-1}^{N}\sum_{m-1}^{M}\sum_{p-1}^{P} z_{ipmn}(p-1)m_n.T.T_{imn}k_k \quad (2)$$

$$\forall i,p,m,n$$
$$Cipm_n - Cip - 1k_k \geq P.T_{imn} + W_{imn} + T.T_{imn}k_h \quad (3)$$
$$\forall i,p,m,n,k,h$$

$$\sum_{n=1}^{N} zipm_n = z_{ipm}, \forall i,p,m. \quad (4)$$

$$zipm_n, Cipm_n \leq C.T_i, \forall i,p,m,n, \quad (5)$$

$$C_{ipm_n} \leq T \quad (6)$$

$$T.Tm_nk_h = \frac{fim_nk_h}{V_{AGV}}, \forall i,k,h,m,m \quad (7)$$

$$z_{ipm_n} \in \{0,1\} \quad (8)$$

Equation 1 is the objective function of the proposed problem which minimizes the material flow. The output of the objective function is the types of the shops which minimize the total material flow. Equation 2 indicates the computation of completion time for each product. Equation 3 certifies that the differences between completion time of product i^{th} in position p^{th} in shop m^{th} of type n^{th} is larger than or equal to the addition of the processing time and waiting time of shop m^{th} of type n^{th} and the transferring time between shop m^{th} of type n^{th} and shop k^{th} of type h^{th}. Equation 4 warranties that if any shop in any position for any product is allocated, then the corresponding shop type is also chosen.

Equation 5 guarantees that if a shop is chosen then the corresponding completion time for each product is lower than or equal to the cycle time. Equation 6 certifies that the completion time for any product in any position in any shop is lower than or equal to the total working time in each day. Equation 7 indicates the transferring time between two shops is related directly to the flow of each product. Equation 8 presents the values of the decision variables.

In order to illustrate the proposed mathematical model, we propose a numerical example. In this example, we consider three products that should be processed in four types of shops each of which is in three shops. Five jobs are existing with respect to their position in the sequence. The material flows between the shops are given in Table 1-5. A large number is assigned to impossible flows. The processing times of each product in the shops are presented in Table 6. The waiting times of each product in the shops are given in Table 7. The shop selection considering the sequence for each product is shown in Table 8.

Table 1: Flows between shops-problem no-1

f	m	n	k	h	f	m	n	k	h	F	m	n	k	h
100,000	1	1	1	1	19	2	2	1	1	13	3	3	1	1
100,000	1	1	1	2	23	2	2	1	2	17	3	3	1	2
100,000	1	1	1	3	11	2	2	1	3	5	3	3	1	3
5	1	1	2	1	100,000	2	2	2	1	13	3	3	2	1
19	1	1	2	2	100,000	2	2	2	2	5	3	3	2	2
100,000	1	1	2	3	100,000	2	2	2	3	100,000	3	3	2	3
10	1	1	3	1	25	2	2	3	1	100,000	3	3	3	1
29	1	1	3	2	17	2	2	3	2	100,000	3	3	3	2
19	1	1	3	3	5	2	2	3	3	100,000	3	3	3	3
17	1	1	4	1	29	2	2	4	1	23	3	3	4	1
25	1	1	4	2	5	2	2	4	2	11	3	3	4	2
100,000	1	1	4	3	100,000	2	2	4	3	100,000	3	3	4	3
100,000	1	2	1	1	100,000	2	3	1	1	17	4	1	1	1
100,000	1	2	1	2	100,000	2	3	1	2	5	4	1	1	2
100,000	1	2	1	3	100,000	2	3	1	3	17	4	1	1	3
17	1	2	2	1	100,000	2	3	2	1	11	4	1	2	1
23	1	2	2	2	100,000	2	3	2	2	29	4	1	2	2
100,000	1	2	2	3	100,000	2	3	2	3	100,000	4	1	2	3
11	1	2	3	1	100,000	2	3	3	1	5	4	1	3	1
5	1	2	3	2	100,000	2	3	3	2	11	4	1	3	2
17	1	2	3	3	100,000	2	3	3	3	23	4	1	3	3
5	1	2	4	1	100,000	2	3	4	1	100,000	4	1	4	1
32	1	2	4	2	100,000	2	3	4	2	100,000	4	1	4	2
100,000	1	2	4	3	100,000	2	3	4	3	100,000	4	1	4	3
100,000	1	3	1	1	11	3	1	1	1	25	4	2	1	1
100,000	1	3	1	2	11	3	1	1	2	29	4	2	1	2
100,000	1	3	1	3	11	3	1	1	3	17	4	2	1	3
22	1	3	2	1	5	3	1	2	1	22	4	2	2	1
11	1	3	2	2	25	3	1	2	2	5	4	2	2	2
100,000	1	3	2	3	100,000	3	1	2	3	100,000	4	2	2	3
13	1	3	3	1	100,000	3	1	3	1	17	4	2	3	1
5	1	3	3	2	100,000	3	1	3	2	23	4	2	3	2
5	1	3	3	3	100,000	3	1	3	3	11	4	2	3	3
17	1	3	4	1	5	3	1	4	1	100,000	4	2	4	1
17	1	3	4	2	31	3	1	4	2	100,000	4	2	4	2
100,000	1	3	4	3	100,000	3	1	4	3	100,000	4	2	4	3
5	2	1	1	1	29	3	2	1	1	100,000	4	3	1	1
17	2	1	1	2	5	3	2	1	2	100,000	4	3	1	2
5	2	1	1	3	5	3	2	1	3	100,000	4	3	1	3
100,000	2	1	2	1	23	3	2	2	1	100,000	4	3	2	1
100,000	2	1	2	2	17	3	2	2	2	100,000	4	3	2	2
100,000	2	1	2	3	100,000	3	2	2	3	100,000	4	3	2	3
5	2	1	3	1	17	3	2	3	1	100,000	4	3	3	1
23	2	1	3	2	100,000	3	2	3	2	100,000	4	3	3	2
19	2	1	3	3	100,000	3	2	3	3	100,000	4	3	3	3
11	2	1	4	1	11	3	2	4	1	100,000	4	3	4	1
25	2	1	4	2	23	3	2	4	2	100,000	4	3	4	2
100,000	2	1	4	3	100,000	3	2	4	3	100,000	4	3	4	3

NSGA II algorithm: The notion of NSGA was first suggested by (Goldberg, 1989) and then implemented by Srinivas and Deb (1994). The main idea behind the non-dominated sorting procedure is that a ranking selection method is used to emphasize good points and a niching method is used to maintain a stable subpopulation of good points. NSGA differs from a simple genetic algorithm only in the way to select operator works. The crossover and mutation operators remain as usual. The efficiency of NSGA lies in the way of multiple objectives is reduced to a single fitness measure by the creation of number of fronts, sorted according to nondomination. Although the NSGA approach has been successfully applied on a number of multi-objective optimization problems, the main criticism of the NSGA approach has been (i) its high computational complexity of non-dominated sorting, O (MN^3) where M is the number of objectives and N is the population size, (ii) the lack of elitism and (iii) the need for specifying the tunable sharing parameter. Recently, Deb *et al*. (2002) reported an improved version of NSGA called NSGA-II to address all of these issues. Specifically, NSGA-II alleviates all the above difficulties by introducing a fast non-dominated sorting procedure with O (MN^2) computational complexity, an elitist-preserving approach and a parameterless niching operator for diversity preservation.

Table 2: Flows between shops problem no 2

f	m	n	k	h	f	m	n	k	h	f	m	n	k	H
100,000	1	1	1	1	18	2	2	1	1	12	3	3	1	1
100,000	1	1	1	2	24	2	2	1	2	16	3	3	1	2
100,000	1	1	1	3	10	2	2	1	3	4	3	3	1	3
4	1	1	2	1	100,000	2	2	2	1	12	3	3	2	1
18	1	1	2	2	100,000	2	2	2	2	4	3	3	2	2
100,000	1	1	2	3	100,000	2	2	2	3	100,000	3	3	2	3
9	1	1	3	1	24	2	2	3	1	100,000	3	3	3	1
28	1	1	3	2	16	2	2	3	2	100,000	3	3	3	2
18	1	1	3	3	54	2	2	3	3	100,000	3	3	3	3
16	1	1	4	1	28	2	2	4	1	22	3	3	4	1
24	1	1	4	2	4	2	2	4	2	10	3	3	4	2
100,000	1	1	4	3	100,000	2	2	4	3	100,000	3	3	4	3
100,000	1	2	1	1	100,000	2	3	1	1	16	4	1	1	1
100,000	1	2	1	2	100,000	2	3	1	2	4	4	1	1	2
100,000	1	2	1	3	100,000	2	3	1	3	16	4	1	1	3
16	1	2	2	1	100,000	2	3	2	1	10	4	1	2	1
22	1	2	2	2	100,000	2	3	2	2	28	4	1	2	2
100,000	1	2	2	3	100,000	2	3	2	3	100,000	4	1	2	3
10	1	2	3	1	100,000	2	3	3	1	4	4	1	3	1
4	1	2	3	2	100,000	2	3	3	2	10	4	1	3	2
16	1	2	3	3	100,000	2	3	3	3	22	4	1	3	3
4	1	2	4	1	100,000	2	3	4	1	100,000	4	1	4	1
32	1	2	4	2	100,000	2	3	4	2	100,000	4	1	4	2
100,000	1	2	4	3	100,000	2	3	4	3	100,000	4	1	4	3
100,000	1	3	1	1	10	3	1	1	1	24	4	2	1	1
100,000	1	3	1	2	10	3	1	1	2	28	4	2	1	2
100,000	1	3	1	3	10	3	1	1	3	16	4	2	1	3
21	1	3	2	1	4	3	1	2	1	21	4	2	2	1
10	1	3	2	2	24	3	1	2	2	4	4	2	2	2
100,000	1	3	2	3	100,000	3	1	2	3	100,000	4	2	2	3
12	1	3	3	1	100,000	3	1	3	1	16	4	2	3	1
4	1	3	3	2	100,000	3	1	3	2	22	4	2	3	2
4	1	3	3	3	100,000	3	1	3	3	10	4	2	3	3
16	1	3	4	1	4	3	1	4	1	100,000	4	2	4	1
16	1	3	4	2	30	3	1	4	2	100,000	4	2	4	2
100,000	1	3	4	3	100,000	3	1	4	3	100,000	4	2	4	3
4	2	1	1	1	28	3	2	1	1	100,000	4	3	1	1
16	2	1	1	2	4	3	2	1	2	100,000	4	3	1	2
4	2	1	1	3	4	3	2	1	3	100,000	4	3	1	3
100,000	2	1	2	1	22	3	2	2	1	100,000	4	3	2	1
100,000	2	1	2	2	16	3	2	2	2	100,000	4	3	2	2
100,000	2	1	2	3	100,000	3	2	2	3	100,000	4	3	2	3
4	2	1	3	1	16	3	2	3	1	100,000	4	3	3	1
22	2	1	3	2	100,000	3	2	3	2	100,000	4	3	3	2
18	2	1	3	3	100,000	3	2	3	3	100,000	4	3	3	3
10	2	1	4	1	10	3	2	4	1	100,000	4	3	4	1
24	2	1	4	2	22	3	2	4	2	100,000	4	3	4	2
100,000	2	1	4	3	100,000	3	2	4	3	100,000	4	3	4	3

Yang and Natarajan (2010) made an attempt to solve multi-objective optimization problem in turning by using NSGA-II. Cheng *et al.* (2009) applied NSGA-II to minimize the comprehensive cost and the whole production load with time-sequence constraints to acquire optimal Collaborative Manufacturing Chain (CMC).

Proposed GA using jumping genes for AGV based FJMS: In this study, NSGA II has been designed to optimize the material flow and makespan in an AGV based jobshop manufacturing system. NSGA II is illustrated in Fig. 2. Initially, a random parent population P_i is created. The population is sorted based on the non-domination. Each solution is assigned a fitness equal to its non-domination level (1 is the best level, 2 is the next-best level and so on). Thus, minimization of fitness is assumed.

At first, the usual selection, crossover and mutation operators are used to create a child population Q_i of size N. Since elitism is introduced by comparing current population with previously-found best non-dominated solutions, a combined population $R_i = P_iUQ_i$ is formed. The population R_i will be of size 2N. Then, the population R_i is sorted according to non-domination. Since all the previous and current population members are included in R_i, the elitism is ensured. Now, solutions belonging to the best non-dominated set F_1 are of best solutions in the combined population and must be emphasized more than any other solution in the combined population. Chromosomes of the front F_1 are included in the new population P_{i+1}. The remaining chromosomes of the population P_{i+1} is chosen from subsequent non-dominated fronts in the order of their

ranking. Thus, solutions from the front F_2 are chosen next, followed by solutions from the front F_3 and so on. This procedure is continued till no more chromosomes can be accommodated. To choose exactly N population chromosomes, the solutions of the last front using the crowded comparison operator α_n, in the descending order and the best solutions needed to fill the population are chosen. The new population $P_{i+1}$is now used for selection, crossover and mutation to create a new population Q_{i+1} of size N.

Input module: The input data required are:

- Number of products
- Number of shop types
- Number of shops
- The material floes between shops

For the experimental problem considered in this study:

- Number of products = 3
- Number of shop types = 3
- Number of shops = 4
- The material flows between shops as given in Table 1-5
- velocity of AGV 4,5,6,7,8,9m/sfor the respective problems

Table 3: Flows between shops problem no 3

f	m	n	k	h	f	m	n	k	h	F	m	n	k	h
100,000	1	1	1	1	25	2	2	1	1	14	3	3	1	1
100,000	1	1	1	2	30	2	2	1	2	20	3	3	1	2
100,000	1	1	1	3	15	2	2	1	3	8	3	3	1	3
8	1	1	2	1	100,000	2	2	2	1	18	3	3	2	1
25	1	1	2	2	100,000	2	2	2	2	8	3	3	2	2
100,000	1	1	2	3	100,000	2	2	2	3	100,000	3	3	2	3
11	1	1	3	1	30	2	2	3	1	100,000	3	3	3	1
31	1	1	3	2	20	2	2	3	2	100,000	3	3	3	2
25	1	1	3	3	56	2	2	3	3	100,000	3	3	3	3
20	1	1	4	1	35	2	2	4	1	24	3	3	4	1
30	1	1	4	2	8	2	2	4	2	15	3	3	4	2
100,000	1	1	4	3	100,000	2	2	4	3	100,000	3	3	4	3
100,000	1	2	1	1	100,000	2	3	1	1	20	4	1	1	1
100,000	1	2	1	2	100,000	2	3	1	2	8	4	1	1	2
100,000	1	2	1	3	100,000	2	3	1	3	20	4	1	1	3
20	1	2	2	1	100,000	2	3	2	1	15	4	1	2	1
24	1	2	2	2	100,000	2	3	2	2	35	4	1	2	2
100,000	1	2	2	3	100,000	2	3	2	3	100,000	4	1	2	3
15	1	2	3	1	100,000	2	3	3	1	8	4	1	3	1
8	1	2	3	2	100,000	2	3	3	2	15	4	1	3	2
20	1	2	3	3	100,000	2	3	3	3	26	4	1	3	3
8	1	2	4	1	100,000	2	3	4	1	100,000	4	1	4	1
34	1	2	4	2	100,000	2	3	4	2	100,000	4	1	4	2
100,000	1	2	4	3	100,000	2	3	4	3	100,000	4	1	4	3
100,000	1	3	1	1	15	3	1	1	1	30	4	2	1	1
100,000	1	3	1	2	15	3	1	1	2	35	4	2	1	2
100,000	1	3	1	3	15	3	1	1	3	20	4	2	1	3
27	1	3	2	1	8	3	1	2	1	27	4	2	2	1
15	1	3	2	2	30	3	1	2	2	8	4	2	2	2
100,000	1	3	2	3	100,000	3	1	2	3	100,000	4	2	2	3
18	1	3	3	1	100,000	3	1	3	1	20	4	2	3	1
8	1	3	3	2	100,000	3	1	3	2	26	4	2	3	2
8	1	3	3	3	100,000	3	1	3	3	15	4	2	3	3
20	1	3	4	1	8	3	1	4	1	100,000	4	2	4	1
20	1	3	4	2	37	3	1	4	2	100,000	4	2	4	2
100,000	1	3	4	3	100,000	3	1	4	3	100,000	4	2	4	3
8	2	1	1	1	35	3	2	1	1	100,000	4	3	1	1
20	2	1	1	2	8	3	2	1	2	100,000	4	3	1	2
8	2	1	1	3	8	3	2	1	3	100,000	4	3	1	3
100,000	2	1	2	1	26	3	2	2	1	100,000	4	3	2	1
100,000	2	1	2	2	20	3	2	2	2	100,000	4	3	2	2
100,000	2	1	2	3	100,000	3	2	2	3	100,000	4	3	2	3
8	2	1	3	1	20	3	2	3	1	100,000	4	3	3	1
26	2	1	3	2	100,000	3	2	3	2	100,000	4	3	3	2
25	2	1	3	3	100,000	3	2	3	3	100,000	4	3	3	3
15	2	1	4	1	15	3	2	4	1	100,000	4	3	4	1
30	2	1	4	2	26	3	2	4	2	100,000	4	3	4	2
100,000	2	1	4	3	100,000	3	2	4	3	100,000	4	3	4	3

Table 4: Flows between shops-problem4

f	m	n	k	h	f	m	n	k	h	f	m	n	k	h
100,000	1	1	1	1	20	2	2	1	1	14	3	3	1	1
100,000	1	1	1	2	24	2	2	1	2	18	3	3	1	2
100,000	1	1	1	3	12	2	2	1	3	6	3	3	1	3
6	1	1	2	1	100,000	2	2	2	1	14	3	3	2	1
20	1	1	2	2	100,000	2	2	2	2	6	3	3	2	2
100,000	1	1	2	3	100,000	2	2	2	3	100,000	3	3	2	3
12	1	1	3	1	26	2	2	3	1	100,000	3	3	3	1
30	1	1	3	2	18	2	2	3	2	100,000	3	3	3	2
20	1	1	3	3	6	2	2	3	3	100,000	3	3	3	3
18	1	1	4	1	30	2	2	4	1	24	3	3	4	1
26	1	1	4	2	6	2	2	4	2	12	3	3	4	2
100,000	1	1	4	3	100,000	2	2	4	3	100,000	3	3	4	3
100,000	1	2	1	1	100,000	2	3	1	1	18	4	1	1	1
100,000	1	2	1	2	100,000	2	3	1	2	6	4	1	1	2
100,000	1	2	1	3	100,000	2	3	1	3	18	4	1	1	3
18	1	2	2	1	100,000	2	3	2	1	12	4	1	2	1
24	1	2	2	2	100,000	2	3	2	2	30	4	1	2	2
100,000	1	2	2	3	100,000	2	3	2	3	100,000	4	1	2	3
12	1	2	3	1	100,000	2	3	3	1	6	4	1	3	1
6	1	2	3	2	100,000	2	3	3	2	12	4	1	3	2
18	1	2	3	3	100,000	2	3	3	3	24	4	1	3	3
6	1	2	4	1	100,000	2	3	4	1	100,000	4	1	4	1
33	1	2	4	2	100,000	2	3	4	2	100,000	4	1	4	2
100,000	1	2	4	3	100,000	2	3	4	3	100,000	4	1	4	3
100,000	1	3	1	1	12	3	1	1	1	26	4	2	1	1
100,000	1	3	1	2	12	3	1	1	2	30	4	2	1	2
100,000	1	3	1	3	12	3	1	1	3	18	4	2	1	3
23	1	3	2	1	6	3	1	2	1	23	4	2	2	1
12	1	3	2	2	26	3	1	2	2	6	4	2	2	2
100,000	1	3	2	3	100,000	3	1	2	3	100,000	4	2	2	3
14	1	3	3	1	100,000	3	1	3	1	18	4	2	3	1
6	1	3	3	2	100,000	3	1	3	2	24	4	2	3	2
6	1	3	3	3	100,000	3	1	3	3	12	4	2	3	3
18	1	3	4	1	6	3	1	4	1	100,000	4	2	4	1
18	1	3	4	2	32	3	1	4	2	100,000	4	2	4	2
100,000	1	3	4	3	100,000	3	1	4	3	100,000	4	2	4	3
6	2	1	1	1	30	3	2	1	1	100,000	4	3	1	1
18	2	1	1	2	6	3	2	1	2	100,000	4	3	1	2
6	2	1	1	3	6	3	2	1	3	100,000	4	3	1	3
100,000	2	1	2	1	24	3	2	2	1	100,000	4	3	2	1
100,000	2	1	2	2	18	3	2	2	2	100,000	4	3	2	2
100,000	2	1	2	3	100,000	3	2	2	3	100,000	4	3	2	3
6	2	1	3	1	18	3	2	3	1	100,000	4	3	3	1
24	2	1	3	2	100,000	3	2	3	2	100,000	4	3	3	2
20	2	1	3	3	100,000	3	2	3	3	100,000	4	3	3	3
12	2	1	4	1	12	3	2	4	1	100,000	4	3	4	1
26	2	1	4	2	24	3	2	4	2	100,000	4	3	4	2
100,000	2	1	4	3	100,000	3	2	4	3	100,000	4	3	4	3

Initialization module: The geno style coding is used to represent the solutions of the problem as chromosomes. Every chromosome consists of three substrings. Each substring consists of 8 bits, the genes of the chromosomes. Every two bits in a substring represents a number between 1-3 (i.e., example 1 is represented as 01, two is represented as 10 and three is represented as 11). First substring represents the shop types for the given positions for product one. Similarly, second and third substring represents the shop types for the given positions for product two and three respectively.

Evaluation module: The objective function is to minimize the material flow and makespan in the AGV based flexible jobshop manufacturing system. The material flow is calculated by using Eq. 1 and makespan is calculated by using Eq. 2.

Non-dominated sorting: This module sorts the population based on non-domination. To start with, the first solution from the population is kept in an empty set P'. Thereafter, each solution p (the second solution onwards) is compared with all members of the set P' one by one. If solution p is dominated by any member of P', the solution p is ignored. If solution p is not dominated by any member of P', it is entered in P'. This is how the set P' grows with non-dominated solutions. When all solutions of the population is checked, the remaining members of P' constitute the non-dominated front. This is illustrated in the Fig. 3.

Measurement of crowding distance: To get an estimate of the density of solutions surrounding a particular solution in the population, the average distance of two points on either side of this point along each of the objectives is calculated. The crowding point computation requires sorting of the population

according to each objective function value in its ascending order of magnitude. Infinite distance is assigned to boundary values for each individual in a particular front. All the remaining solutions are assigned a distance value equal to the absolute difference in the function value of two adjacent solutions. This calculation is continued with other objective functions. The overall crowding distance value is calculated as the sum of individual distance values corresponding to each objective. Crowding distance is calculated by the following formula:

Here:
m(j+1) = Objective function value of $(j+1)^{th}$ individual
m(j-1 = Objective function value of $(j-1)^{th}$ individual
m_{max} = Maximum value of objective function in the front
m_{min} = Minimum value of objective function in the front

The basic idea behind the crowding distance is finding the Euclidian distance between each individual in a front based on their objectives. The individual in the boundary are always selected since they have infinite distance assignment. The crowding distance of the chromosomes in initial population is found. The Fig. 4 shows the sorting process of crowding distance.

Table 5: Flows between shops problem 5

f	m	n	k	h	f	m	n	k	h	f	M	n	k	h
100,000	1	1	1	1	20	2	2	1	1	14	3	3	1	1
100,000	1	1	1	2	24	2	2	1	2	18	3	3	1	2
100,000	1	1	1	3	12	2	2	1	3	6	3	3	1	3
6	1	1	2	1	100,000	2	2	2	1	14	3	3	2	1
20	1	1	2	2	100,000	2	2	2	2	6	3	3	2	2
100,000	1	1	2	3	100,000	2	2	2	3	100,000	3	3	2	3
12	1	1	3	1	26	2	2	3	1	100,000	3	3	3	1
30	1	1	3	2	18	2	2	3	2	100,000	3	3	3	2
20	1	1	3	3	6	2	2	3	3	100,000	3	3	3	3
18	1	1	4	1	30	2	2	4	1	24	3	3	4	1
26	1	1	4	2	6	2	2	4	2	12	3	3	4	2
100,000	1	1	4	3	100,000	2	2	4	3	100,000	3	3	4	3
100,000	1	2	1	1	100,000	2	3	1	1	18	4	1	1	1
100,000	1	2	1	2	100,000	2	3	1	2	6	4	1	1	2
100,000	1	2	1	3	100,000	2	3	1	3	18	4	1	1	3
18	1	2	2	1	100,000	2	3	2	1	12	4	1	2	1
24	1	2	2	2	100,000	2	3	2	2	30	4	1	2	2
100,000	1	2	2	3	100,000	2	3	2	3	100,000	4	1	2	3
12	1	2	3	1	100,000	2	3	3	1	6	4	1	3	1
6	1	2	3	2	100,000	2	3	3	2	12	4	1	3	2
18	1	2	3	3	100,000	2	3	3	3	24	4	1	3	3
6	1	2	4	1	100,000	2	3	4	1	100,000	4	1	4	1
33	1	2	4	2	100,000	2	3	4	2	100,000	4	1	4	2
100,000	1	2	4	3	100,000	2	3	4	3	100,000	4	1	4	3
100,000	1	3	1	1	12	3	1	1	1	26	4	2	1	1
100,000	1	3	1	2	12	3	1	1	2	30	4	2	1	2
100,000	1	3	1	3	12	3	1	1	3	18	4	2	1	3
23	1	3	2	1	6	3	1	2	1	23	4	2	2	1
12	1	3	2	2	26	3	1	2	2	6	4	2	2	2
100,000	1	3	2	3	100,000	3	1	2	3	100,000	4	2	2	3
14	1	3	3	1	100,000	3	1	3	1	18	4	2	3	1
6	1	3	3	2	100,000	3	1	3	2	24	4	2	3	2
6	1	3	3	3	100,000	3	1	3	3	12	4	2	3	3
18	1	3	4	1	6	3	1	4	1	100,000	4	2	4	1
18	1	3	4	2	32	3	1	4	2	100,000	4	2	4	2
100,000	1	3	4	3	100,000	3	1	4	3	100,000	4	2	4	3
6	2	1	1	1	30	3	2	1	1	100,000	4	3	1	1
18	2	1	1	2	6	3	2	1	2	100,000	4	3	1	2
6	2	1	1	3	6	3	2	1	3	100,000	4	3	1	3
100,000	2	1	2	1	24	3	2	2	1	100,000	4	3	2	1
100,000	2	1	2	2	18	3	2	2	2	100,000	4	3	2	2
100,000	2	1	2	3	100,000	3	2	2	3	100,000	4	3	2	3
6	2	1	3	1	18	3	2	3	1	100,000	4	3	3	1
24	2	1	3	2	100,000	3	2	3	2	100,000	4	3	3	2
20	2	1	3	3	100,000	3	2	3	3	100,000	4	3	3	3
12	2	1	4	1	12	3	2	4	1	100,000	4	3	4	1
26	2	1	4	2	24	3	2	4	2	100,000	4	3	4	2
100,000	2	1	4	3	100,000	3	2	4	3	100,000	4	3	4	3

Table 6: Processing times

Product one				Product two				Product three			
	n			n				n			
Problem no1											
m	1	2	3	m	1	2	3	m	1	2	3
1	6	3	4	1	7	7	4	1	6	8	6
2	5	8	1,000	2	8	6	1,000	2	5	9	1,000
3	10	9	6	4	11	13	4	3	8	12	8
4	12	14	1,000	4	12	7	1,000	4	7	11	1,000
Problem no2											
m	1	2	3	m	1	2	3	M	1	2	3
1	1	1	2	1	4	4	2	1	3	3	2
2	3	4	1,000	2	4	3	1,000	2	4	5	1,000
3	1	5	5	3	1	3	3	3	3	6	3
4	2	4	1,000	4	2	4	1,000	4	3	1	1,000
Problem no3											
m	1	2	3	m	1	2	3	M	1	2	3
1	2	3	1	1	2	6	3	1	2	2	4
2	3	2	1,000	2	1	3	1,000	2	1	1	1,000
3	2	4	3	3	6	1	2	3	4	4	2
4	3	2	1,000	4	3	2	1,000	4	2	2	1,000
Problem no 4											
m	1	2	3	m	1	2	3	M	1	2	3
1	4	4	3	1	2	2	4	1	2	6	5
2	5	5	1,000	2	2	4	1,000	2	2	2	1,000
3	4	3	1	3	3	1	5	3	2	7	7
4	2	3	1,000	4	1	2	1,000	4	2	3	1,000
Problem no 5											
m	1	2	3	m	1	2	3	M	1	2	3
1	7	6	3	1	1	2	4	1	2	3	5
2	6	4	1,000	2	2	1	1,000	2	3	2	1,000
3	3	3	1	3	1	1	2	3	3	2	7
4	2	2	1,000	4	2	3	1,000	4	2	2	1,000

Selection module: This module is constructed on the basis of tournament selection mechanism. The size of the mating pool N is filled by randomly choosing n chromosomes for each individual in the population. One chromosome out of n is selected based on the rank and crowding distance. That is, the chromosome with least rank is selected. In case of more chromosomes having same rank, the chromosome having highest crowding distance is selected. This selected chromosome is added to the mating pool. Thus, tournament selection is done to generate the mating pool based on the rank and the crowding distance:

$$d = d(0) + \frac{m(j+1) - m(j-1)}{m_{max} - m_{min}}$$

In this proposed algorithm n value is taken as 3. Therefore, for every chromosome in the new mating pool, three chromosomes are randomly chosen from the population. Out of three, one chromosome is selected based on rank and crowding distance value.

Jumping genes module: Jumping operatiors has unique feature depends upon chromosome. In our study two genes at different positions are randomly selected. The equal number of genes selected from the other chromosomes at random position. The two selected chromosomes are transferred mutually each other. During jumping genes operation each chromosome consists transposons. These are having the tendancy to jump to other chromosomes. The number of transposons canbe more than one in general case but, inour case it is two. The locations of transposon are randomly chosen. The cut and paste method is used to implement the jumping gene operation. The element is cut from the original site and pasted into a new location in our case two genes are consider as transposons.The jumping genes operation is illustrated in Table 9. In this example the last position of parent one is selected and gene is jumped last position of second parent chromosome and vice versa.

Table 7: Waiting time

Product one				Product two				Product three			
	n			*n*				*n*			
Problem no1											
m	1	2	3	m	1	2	3	m	1	2	3
1	2	4	5	1	1	3	3	1	2	2	1
2	4	6	1,000	2	3	2	1,000	2	5	3	1,000
3	2	8	3	3	4	1	4	3	2	2	
4	1	3	1,000	4	3	1	1,000	4	5	11	1,000
Problem no2											
	n			n				n			
m	1	2	3	m	1	2	3	m	1	2	3
1	2	3	2	1	3	8	7	1	3	3	9
2	8	2	1,000	2	6	8	1,000	2	7	9	1,000
3	9	6	7	3	4	5	6	3	6	6	4
4	10	7	1,000	4	2	9	1,000	4	3	5	1,000
Problem no3											
		n			n			n			
m	1	2	3	m	1	2	3	m	1	2	3
1	2	3	1	1	2	6	3	1	2	2	4
2	3	2	1,000	2	1	3	1,000	2	1	1	1,000
3	2	4	3	3	6	1	2	3	4	4	2
4	3	2	1,000	4	3	2	1,000	4	2	2	1,000
Problem no 4											
	n			n			n				
m	1	2	3	m	1	2	3	m	1	2	3
1	4	4	3	1	2	2	4	1	2	6	5
2	5	5	1,000	2	2	4	1,000	2	2	2	1,000
3	4	3	1	3	3	1	5	3	2	7	7
4	2	3	1,000	4	1	2	1,000	4	2	3	1,000
Problem no 5											
		n			n			n			
m	1	2	3	m	1	2	3	m	1	2	3
1	7	6	3	1	1	2	4	1	2	3	5
2	6	4	1,000	2	2	1	1,000	2	3	2	1,000
3	3	3	1	3	1	1	2	3	3	2	7
4	2	2	1,000	4	2	3	1,000	4	2	2	1,000

Crossover module: In the population obtained through the selection module, the crossover operation combines two good chromosomes to hopefully form two better chromosomes.

Every crossover may not create better solutions and if bad solutions are created they will get eliminated in the next selection operation due to lower fitness value. Again, in order to preserve some good chromosomes selected during the selection operation, not all the chromosomes in the population are used in the crossover. Hence crossover operation is exercised on the chromosomes of the population with a probability, known as cross over probability (p_c). Crossover module consists of the following two steps.

Step 1: Selection of chromosome for crossover.

Selection of chromosomes for crossover is done with a crossover probability p_c and to choose an appropriate crossover probability a sensitivity analysis is conducted. The algorithm is executed with ten different initial seeds for different combinations of genetic algorithm parameters. After doing the sensitivity analysis, it is found that minimum material flow and makespan are attainable for a crossover probability, p_c of 0.6. Therefore, 60% of the chromosomes are selected to undergo crossover operation. Random number r between zero and one are generated for all chromosomes. If the generated random number r is less than 0.6 then the corresponding chromosome is selected for crossover operation.

Step 2: Crossover operation

The chromosomes selected for crossover are mated pair wise and undergo two point crossover operation. The crossover-site genes are chosen by creating two random numbers between 1 and 50. An illustrative example of crossover operation between two chromosomes is

exhibited in Table 10. In this example the 7th and 12th genesare randomly selected as the crossover site. The genes within the crossover site of the parent chromosomes are crossed to produce the two offsprings.

Mutation module: The mutation operator alters genes of a chromosome locally to hopefully create a better chromosome. It is expected that if bad chromosomes are created they will be eliminated by the selection operation in subsequent generations and if good solutions are created, they will be emphasized. The need for mutation is to create a point in the neighbourhood of the current point, thereby achieving a local search around the current solution. The mutation is also used to maintain the diversity in the population. The mutation operator has a constructive as well as destructive effect. As it can create a better solution by perturbing a solution, it can also destroy a good solution.

Table 8: Sequences

Product 1					Product 2					Product 3				
	n				n					n				
Problem no 1														
M	1	2	3	4	m	1	2	3	4	m	1	2	3	4
1	1	0	0	0	1	0	1	0	0	1	1	0	0	0
2	0	1	0	0	2	1	0	0	0	2	0	1	0	0
3	0	0	1	0	3	0	0	1	0	3	0	0	0	1
4	0	0	0	1	4	0	0	0	1	4	0	0	1	0
Problem no 2														
M	1	2	3	4	m	1	2	3	4	m	1	2	3	4
1	0	0	0	1	1	0	1	0	0	1	1	0	0	0
2	0	1	0	0	2	0	0	0	1	2	0	0	1	0
3	0	0	1	0	3	0	0	1	0	3	0	1	0	0
4	1	0	0	0	4	1	0	0	0	4	0	0	0	1
Problem no 3														
M	1	2	3	4	m	1	2	3	4	m	1	2	3	4
1	0	0	0	1	1	0	0	1	0	1	1	0	0	0
2	0	1	0	0	2	0	0	0	1	2	0	1	0	0
3	1	0	0	0	3	0	1	0	0	3	0	0	1	0
4	0	0	1	0	4	1	0	0	0	4	0	0	0	1
M	1	2	3	4	m	1	2	3	4	m	1	2	3	4
1	1	0	0	0	1	0	0	1	0	1	0	1	0	0
2	0	1	0	0	2	0	1	0	0	2	1	0	0	0
3	0	0	0	1	3	1	0	0	0	3	0	0	1	0
4	0	0	1	0	4	0	0	0	1	4	0	0	0	1
Problem no 5														
M	1	2	3	4	m	1	2	3	4	m	1	2	3	4
1	0	1	0	0	1	0	0	1	0	1	1	0	0	0
2	0	0	0	1	2	0	1	0	0	2	0	0	1	0
3	0	0	1	0	3	1	0	0	0	3	0	0	0	1
4	1	0	0	0	4	0	0	0	1	4	0	1	0	0

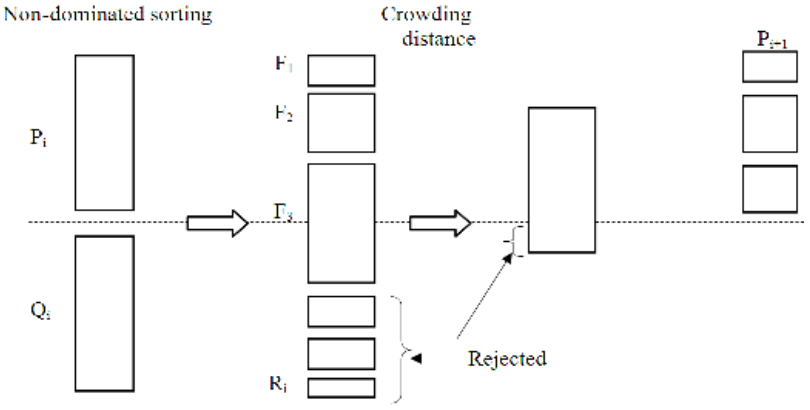

Fig. 2: A sketch of NSGA-II

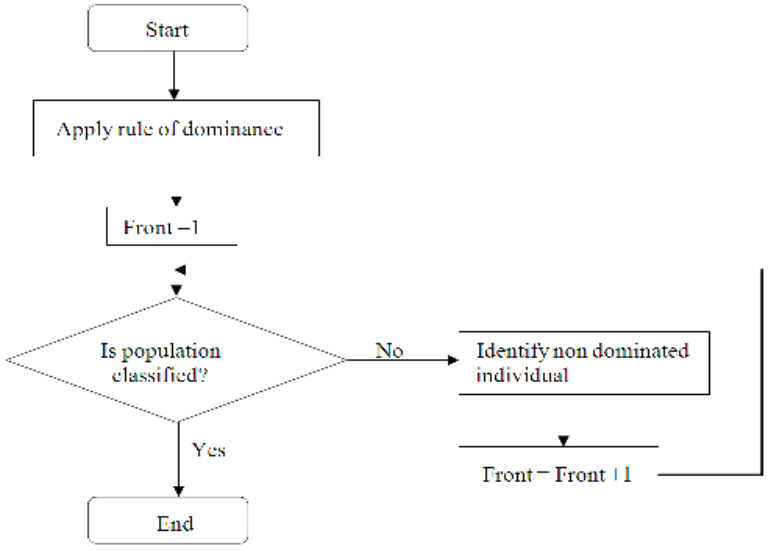

Fig. 3: Flowchart for non-dominated sorting

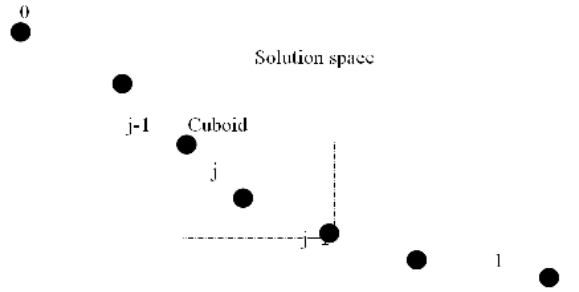

Fig. 4: Crowding distance calculation between pareto solutions in a solution space

As it is preferred in accepting the constructive effect and since it is computationally expensive to check the worth of every possible mutation for its outcome, the mutation is generally used with a small probability (p_m). It is found that the minimum material flow and makespan are attainable for a mutation probability p_m of 0.05. Hence, p_m is taken as 0.05. In this problem, the mutation is carried out for all genes. A random number is generated corresponding to all genes in the population. If the random number r is less than 0.05, then that corresponding gene is mutated. In mutation, the gene value 0 is converted into 1 and 1 is converted into 0. The mutation operation is illustrated in Table 11.

Generation of intermediate population: In NSGA-II, an intermediate population is formed which is the combined population of parents and offsprings of the current generation. Naturally, the resultant population size is greater than the original population size.

Table 9: Chromosomes before and after jumping operation

Operation		Chromosome with three substring (gene position)		
Before jumping	Parent I	11101110	1101110	10010010
Operation	parent II	111011	10101111	11011001
After jumping	Parent I	11101100	1111110	1010010
Operation	Parent II	10111011	10101101	11011010

Table 10: Chromosomes before and after crossover

Operation		Chromosome with three substring (gene position)		
Before	Parent I	11101110	01101110	10010010
Cross over	parent II	00111011	10101111	11011001
After	Parent I	11101111	10101110	10010010
Cross over	Parent II	00111010	01101111	11011001

Table 11: Chromosomes before and after mutation

Operation		Chromosomes with three substrings each (Gene Positions)$_1$
Before mutation	Original chromosome	11011001 01100111 10010001
After mutation	Mutated chromosome	11010001 01100011 10110001

Non-dominated sorting is performed again on the intermediate population since potential chromosomes may be present both in the parent and the offspring populations. New population is formed by replacing chromosomes in the original population. The chromosomes with higher ranks are selected and added to the population until the population size is reached. The last front is included in the population based on the crowding distance.

Termination criterion: The NSGA-II is terminated when the iteration number reaches its maximum value. In this algorithm, 500 is taken as the maximum number of iteration.

RESULTS

The proposed jumping genes GA algorithm designed for the AGV based flexible jobshop manufacturing system, is coded in MATLAB and the experiments were conducted on an Intel Core2 Duo Processor computer.

The proposed algorithm is tested on five example problems. The proposed algorithm try to minimizes both the material flow time and makespan.

Table 12: Makespan and flow time of respective AGV sequences

Problem	AGV sequence	Makespan	Flow time	Product completion time
1	2-1;1-1;2-1;1-2;2-1;1-3;2-2;1-1;3-3;	57	85.6	142.6
2	2-1;3-3;2-2;3-1;1-1;3-1;4-1;3-3;4-2;3-3;4-11-2;3-1;2-2;4-1;3-1;4-1;2-2;3-1;2-1;	32	110	142
3	1-3;2-2;3-1;4-1;3-1;2-1;4-1;3-1;1-2;3-1;1-1 3-3;4-2;4-1;2-1;4-2;2-2;1-3;2-1;2-	53	41.2	94.2
4	1;3-1;2-2;1-1;2-1;1-3;1-3;3-3;3-1;4-1	51	59.8	110.8
5	2-1;1-3;2-1;1-3;3-2;1-2;2-1;3-3;2-1;3-2;4-2;2-1;1-2;2-1;4-1;1-2;4-1;3-3;1-2;4-1.3-2;2-1;1-1;3-3;2-2;4-1;3-3;4-1;3-1;4-1;3-3;4-1;2-2;1-3;4-1;3-3;1-3	41	63.42	104.42

Table 13: Comarision

			Completion time					
	Flow time		Product 1		Product 2		Product 3	
Problem	Hamed Fazlollah tabaret al	Jumping genes	Hamed Fazlollah tabaret al	Jumping genes	Hamed Fazlollah tabaret al	Jumping genes	Hamed Fazlollah tabaret al	Jumping genes
1	88.6	85.6	61	57	31	31	47	43
2	110	110	38	32	31	28	32	32
3	87	87	46	32	48	48	53	53
4	60.66	59.8	42	42	55	51	34	34
5	63.42	63.66	38	38	44	41	34	34

Table 14: Makespan calculation for simultaneous processing of products

Time	Product one (m-n)	Product two (m-n)	Product three (m-n)
0	1-2	2-1	1-3
7	--	1-1	--
8	2-1	--	--
9	--	--	2-2
15	--	3-3	--
21	--	--	3-1
23	--	4-2	--
31	--		4-1
35	3-3		--
44	4-1	--	--
47			--
57	--	--	--

Table15: Shop types in different positions for three products minimizing both material flow time and makespan

Problem 1

Completion time	m	n	Completion time	m	n	Completion time	m	n
8	1	2	7	2	1	9	1	3
35	2	1	15	1	1	21	2	2
44	3	3	23	3	3	31	3	1
57	4	1	31	4	2	43	4	1

Then for every problem, an experiment is conducted to obtain shop types for products which is used to optimize the material flow time seperately. The obtained result is given in Table 12. For example for illustrating the problem one, to complete this process, the AGV has to follow the sequence 2-1;1-1;2-1;1-2;2-1;1-3;2-2;1-1;3-3;2-1;3-3;2-2;3-1;1-1;3-1;4-1;3-3;4-2;3-3;4-1. The flow time is 85.6 m. Total makespan is 57+85.6 = 142.6 m.

This experiment is repeated again to obtain shop types which minimize the makespan alone. In order to utilize the machine shops effectively, a simultaneous processing of products and the completion time of products are also considered to minimize the total makespan. These results are shown in Table 14-23.

In order to find the efficiency of the jumping genes GA algorithm, the five example problems are experimented using Hamed fazlollahtabar mathematical method and the results are shown in Table 13.

Table 16: Makespan calculation for simultaneous processing of products

Time	Product one (m-n)	Product two (m-n)	Product three (m-n)
0	4-1	2-2	1-2
6	--	--	3-1
12	2-2	4-1	--
15	--	--	2-1
16		3-1--	--
18	3-1		--
21		1-1	--
26			4-1
28	1-2		
32	--		--

Table 17: Shop types in different positions for three products minimizing both material flow time and makespan

Problem 2

Completion time	m	n	Completion time	m	n	Completion time	m	n
12	4	1	12	2	2	6	1	2
18	2	2	16	4	1	15	3	1
28	3	1	21	3	1	26	2	1
32	1	2	28	1	1	32	4	1

Table 18: Makespan calculation for simultaneous processing of products

Time	Product one (m-n)	Product two (m-n)	Product three (m-n)
0	4-1	3-3	1-3
5	--	4-2	--
15	2-1	--	--
19	1-3		
20		2-2	2-1
24	3-3	--	
27		--	3-1
29	--	1-1	--
32			
40		--	--
48	--	--	4-1
53	--	--	

Table 19: Shop types in different positions for three products minimizing both material flow time and makespan

Problem 3								
Completion time	m	n	Completion time	m	n	Completion time	m	n
15	4	1	5	3	3	20	1	3
19	2	1	20	4	2	27	2	1
24	1	3	29	2	2	48	3	1
32	3	3	48	1	1	53	4	1

Table 20: Makespan calculation for simultaneous processing of products

Time	Product one (m-n)	Product two (m-n)	Product three (m-n)
0	1-2	3-3	2-1
4	--	--	1-3
12	--	--	3-2
16	2-1	--	--
26	--		4-2
27	4-1	2-1	
33	3-3		--
34		1-2	--
48		4-1	
51	--		--

Table 21: Shop types in different positions for three products minimizing both material flow time and makespan

Problem 4								
Completion time	m	n	Completion time	m	n	Completion time	m	n
16	1	2	27	3	3	4	2	1
27	2	1	34	2	1	12	1	3
33	4	1	48	1	2	26	3	2
42	3	3	51	4	1	34	4	2

Table 22: Makespan calculation for simultaneous processing of products

Time	Product one (m-n)	Product two (m-n)	Product three (m-n)
0	2-2	3-2	1-1
8	--	2-1	3-3
9	4-1	--	--
11	--	1-3	--
15	--	--	4-1
19	--		--
23	3-3	--	
27	--	4-1	2-2
29	1-3	--	--
38		--	--

Table 23: Shop types in different positions for three products minimizing both material flow time and makespan

Problem 5								
Completion time	m	n	Completion time	m	n	Completion time	m	n
9	2	2	8	3	2	8	1	1
23	4	1	11	2	1	15	3	3
29	3	3	27	1	3	27	4	1
38	1	3	41	4	1	34	2	2

DISCUSSION

The completion time of the product two is minimum for the problem number 2, problem number 4 and problem5 thant problem number 1 and 3.For the product three, the completion time is comparably minimum for the problem number one only.

Almost flowtime and completion time of each product is comparably minimum than Hamed fazlollahtabar method. The completion time of product one is minimum for problem number 1-3 except problem number 4 and 5.

CONCLUSION

In this study, an attempt is made to address the issues related to an AGV based flexible jobshop manufacturing system with the objectives of minimizing the material flow and makespan as a whole. Since it is a nonlinear programming problem, jumping genes GA algorithm is developed to solve this problem.

The proposed algorithm is tested with an AGV based flexible manufacturing system. It is found that the proposed algorithm is able to produce quality solutions yielding minimum material flow time (minutes) and makespan (minutes). The jumping genes GA algorithm results are compared with Hamed fazlollahtabar method.

In order to maximize the machine utilization, a simultaneous processing of products is considered. Here, all the three products are processed simultaneously. The result shows that, the makespan to complete all the products minimized. This algorithm may applied for more number of AGV s. for future study.

REFERENCES

Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by tabu search. Ann. Oper Res., 41: 157-183. DOI: 10.1007/BF02023073

Brucker, P. and R. Schlie, 1990. Job-shop scheduling with multi-purpose machines. Computing, 45: 369-375. DOI: 10.1007/BF02238804

Chen, J.C., K.H. Chen, J.J. Wu and C.W. Chen, 2008. A study of the flexible job shop scheduling problem with parallel machines and reentrant process. Int. J. Adv. Manuf Techno., 39: 344-354. DOI: 10.1007/s00170-007-1227-1

Cheng, F., J. Yang and F. Ye, 2009. Multi-objective optimization of collaborative manufacturing chain with time-sequence constraints. Int. J. Adv. Manuf. Technol., 40: 1024-1032. DOI: 10.1007/s00170-008-1388-6

Deb, K., A. Pratap and S.A. Meyarivan, 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Evolut Comput 6: 182-197. DOI: 10.1109/4235.996017

Fazlollahtabar, H., B. Rezaie and H. Kalantari, 2010. Mathematical programming approach to optimize material flow in an AGV-based flexible jobshop manufacturing system with performance analysis. Int. J. Adv. Manuf. Technol., 51: 1149-1158. DOI: 10.1007/s00170-010-2700-9

Gao, J., M. Gen, L.Y. Sun and X.H. Zhao, 2007. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Comput. Ind. Eng., 53: 149-162. DOI: 10.1016/j.cie.2007.04.010

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. 1st Edn., Addison-Wesley, New York, ISBN-10: 0201157675, pp: 432.

Ho, N.B. and J.C. Tay, 2007. Using evolutionary computation and local search to solve multi-objective flexible job shop problems. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, (GEC' 07), ACM New York, NY, USA, pp: 821-828. DOI: 10.1145/1276958.1277121

Kacem, I., S. Hammadi and P. Borne 2002a. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Math. Comput. Simul., 60: 245-276. DOI: 10.1016/S0378-4754(02)00019-8

Kacem, I., S. Hammadi and P. Borne, 2002b. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Syst. Man Cybern, 32: 1-13. DOI: 10.1109/TSMCC.2002.1009117

Kannaiah, S.K., J. Thangavel, D.P. Kothari, 2011. A genetic algorithm based multi objective service restoraton in distribution system. J. Comput. Sci., 7: 448-453. DOI: 10.3844/jcssp.2011.448.453

Liu, H.B., A. Abraham, O. Choi and S.H. Moon, 2006. Variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems. Proceedings of the 6th International Conference on Simulated Evolution and Learning, (SEL' 06), Springer-Verlag Berlin, Heidelberg, pp: 197-204. DOI: 10.1007/11903697_26

Mansour, M.A.A.F., 2011. A Genetic algorithm for scheduling n jobs on a single machine with stochastic controllable processing, tooling cost and earliness – tardiness penelties. Am. J. Eng. Applied Sci., 4: 341-349. DOI: 10.3844/ajeassp.2011.341.349

Ripon, K.S.N., C.H. Tsang and S. Kwong, 2006. Multi-objective evolutionary job-shop scheduling using jumping genes genetic algorithm. Proceedings of the International Joint Conference on Neural Networks, (IJCNN '06), IEEE Xplore Press, Vancouver, BC., pp: 3100-3107. DOI: 10.1109/IJCNN.2006.247291

Saidi-Mehrabad, M. and P. Fattahi, 2007. Flexible job shop scheduling with tabu search algorithms. Int. J. Adv. Manuf. Technol., 32: 563-570. DOI: 10.1007/s00170-005-0375-4

Srinivas, N. and K. Deb, 1994. Multiobjective optimization using nondominated sorting in genetic algorithm. Evolut. Comput., 2: 221-248. DOI: 10.1162/evco.1994.2.3.221

Tay, J.C. and N.B. Ho, 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput. Indus. Eng., 54: 453-473. DOI: 10.1016/j.cie.2007.08.008

Thmilselvan, R. and P. Balasubramanie, 2012. Intergration of genetic algorithm with tabusearch for jobshop scheduling with unordered subsequence exchange crossover. J. Comput. Sci., 8: 681-693. DOI: 10.3844/jcssp.2012.681.693

Xia, W. and Z. Wu, 2005. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput. Ind. Eng., 48: 409-425. DOI: 10.1016/j.cie.2005.01.018

Xing, L.N., Y.W. Chen and K.W. Yang, 2009. An efficient search method for multi-objective flexible job shop scheduling problems. J. Intell. Manuf., 20: 283-293. DOI: 10.1007/s10845-008-0216-z

Yang, S.H. and U. Natarajan, 2010. Multi-objective optimization of cutting parameters in turning process using differential evolution and non-dominated sorting genetic algorithm-II approaches. Intell. J. Adv. Manuf. Technol., 49: 773-784. DOI: 10.1007/s00170-009-2404-1

Yussof, S., A. Raina and H. Razali, 2011. An Investigation of using parallel genetic algorithm for solving the shortest path routing problem. J. Comput. Sci., 7: 206-215. DOI: 10.3844/jcssp.2011.206.215

Zhang, G., X. Shao, P. Li and L. Gao, 2009. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Comput. Ind. Eng., 56: 1309-1318. DOI: 10.1016/j.cie.2008.07.021