Chapter 28 A Novel Way to Schedule Flexible Manufacturing System



Srushti Bhatt, M. B. Kiran, and Jeetendra A. Vadher

Abstract Modeling and scheduling of flexible manufacturing system problems are yet to be addressed simply to compete in the global manufacture market. Optimizing the performance measure of the available resources of manufacturing systems is a key requirement. Many researchers have proposed techniques for solutions like mathematical programming, heuristic dispatching methods, and artificial intelligence and knowledge base system. In the present work, a combined approach of the Timed Petri Net and genetic algorithm is used for the modeling power and optimization capabilities for scheduling flexible manufacturing system. Petrinet plays a vital role in modeling FMS. The power of Timed Petrinet is used for modeling FMS by using the advantage of its ability to model a complex system with efficient net structure and chromosomal representation along with genetic algorithm for optimal solution needed in scheduling a manufacturing problem. The result obtained by this approach concludes that the proposed method performs much better than the existing methods. The algorithm is developed for minimizing makespan and the simulation run for better performance measures.

Keywords FMS · GA · Petri Nets · SPN

S. Bhatt (🖂)

Department of Mechanical Engineering, Pandit Deendayal Petroleum University, Gandhinagar, Gujarat 382007, India

e-mail: bhattsrushti12@gmail.com

M. B. Kiran

J. A. Vadher Department of Mechanical Engineering, Government Engineering College, Palanpur, Gujarat 385001, India

© Springer Nature Singapore Pte Ltd. 2021 A. Sachdeva et al. (eds.), *Operations Management and Systems Engineering*, Lecture Notes on Multidisciplinary Industrial Engineering, https://doi.org/10.1007/978-981-15-6017-0_28

Department of Industrial Engineering, Pandit Deendayal Petroleum University, Gandhinagar, Gujarat 382007, India

Nomenclature

- FMS Flexible manufacturing system
- GA Genetic algorithm
- PN Petri nets
- TPN Timed Petri nets

28.1 Introduction

In the present market scenario, to stand in a market it is very important for a manufacturing system to accommodate changes quickly as customer demand changes. Many Indian industries have adopted FMS for in time production and quality of the product, but still problems in scheduling still arise. The scheduling problem is nothing but simply a process of resource allocation to accomplish a specified task [1].

Petri nets are a class of modeling tools, which were originated by Petri [2, 3], they have a well-defined mathematical foundation and easy to understand the graphical feature. It is a powerful graphical tool for modeling and analyzing concurrent, parallel, simultaneous, synchronous, distributed, and resource sharing systems with the advantages like an easy visualization of complex systems can model a system hierarchically (a top-down fashion at various levels of abstraction and detail) and can analyze qualitative and quantitative aspects of the system.

The common definition of PNs introduced by Petri is as follows. Petri nets or place/transition net can be defined as a five-tuple: PN = (P, T, I, O, Mo), where P and T are finite non-empty sets of places pictured by circles and transitions pictured by bars, respectively. I: $P \times T \rightarrow \{0, 1\}$ is an input function that defines the set of directed arcs from P to T. O: $P \times T \rightarrow \{0, 1\}$ is an output places which are drawn as circles represent possible states or conditions of the system while transition, which is shown in bars or boxes, describe events that may modify the system states. The relationships between places and transitions are represented by a set of arcs that are the only connectors between a place and a transition in either direction. The dynamic behavior of the system can be represented using tokens which graphically appear as black dots in places. A transition can only fire if it is enabled and having one token in that place I to the next transition j. A timed Petri net is associated with places or transitions. Here we consider the time associated with places. The modeling of the) where M0 is initial marking, Mr0 is initial vector for remaining processes, is set of time delays associated with it. The problem data consist of job-based data, operationbased data, and machine data including processing time or operation time along with the path of the operation for the given jobs. Different researchers have used different approaches of Petri nets for simulation and GA for fine-tuning the parameters [2]. Different researchers have identified different criteria for solving FMS scheduling problem. One of them is performance measures. The list of performance criteria are

S. No.	Criteria	Code
1	Idle time	T idle
2	Length of the AGVs route	Route length
3	Number of backtrackings of each AGV	Backtrackings
4	Total flow time	F
5	Mean flow time	F medium
6	Maximum lateness	L _{max}
7	Makespan	C _{max}
8	Tardiness	Т
9	Maximum tardiness	T _{max}
10	Due date	D _d
11	Cost for tardiness and earliness; production cost, penalty cost	Cost
12	Throughput	Т
13	Work in process	WIP
14	Machine utilization	U
15	Maximum utilization of the machines	U _{max}

Table 28.1 Criteria for performance parameters (Reproduced from Filho et al. [4])

as stated in Table 28.1. Here, out of all performance parameters attempt is made to address the makespan using a hybrid approach combining Petri nets and GA.

28.2 **Problem Definition**

The makespan minimization problem of a flexible manufacturing system (FMS) has been recognized as one of the most important planning problems. Job scheduling problems are referred to as NP-hard problems. And to solve such kind of problem there are different four types of representation. They are job-based representation, operation-based representation, priority rule-based, and preference-list-based representation [1].

In this research, a Genetic Algorithm (GA) based heuristic is proposed to solve the makespan of a random type FMS. The objective of the problems is to minimize the system unbalance and maximize the throughput, satisfying the technological constraints such as availability of machining time, and tool slots. The proposed GA-based heuristic determines the part type sequence and the operation-machine allocation that guarantee the optimal solution to the problem, rather than using fixed predetermined part sequencing rules.

The first objective of the research work is to schedule the given FMS, modeling with Petri nets and optimizing using genetic algorithm and obtaining results to conclude with the best option. The makespan minimization problem consists of three types Job shop scheduling (there are n jobs and m identical stations. Each job should

Job	Operation tin	ne		Operation path						
	M/C 1	M/C 2	M/C 3	M/C 1	M/C 2	M/C 3				
1	4	3	2	1	2	3				
2	1	4	4	2	1	3				
3	3	2	3	3	2	1				
4	3	3	1	2	3	1				

Table 28.2 Problem data

be executed on a single machine), Open shop scheduling (there are n jobs and m different stations. Each job should spend some time at each station, in a free order), Flow shop scheduling (there are n jobs and m different stations. Each job should spend some time at each station, in a predetermined order). To optimize makespan the system consists of the following:

The system considered is

n = four jobs and m = three machines with the respective operating time. Each job is processed on every machine at the appropriate time.

The machine can operate only one operation at a time.

All machines are available.

Here one job will leave the machine only after completing the operation and scheduling is done in the way that no classes with the same job will assign to two different machines at the same time or vice versa.

Machine setup time is included in the processing time.

The data of n jobs and m machines are as given in Table 28.2.

28.3 Methodology

For the given data will be evaluated into two phases. The first phase gives the system model and the second gives the optimum result.

Step 1: Creating a system model using Petri net tool using MATLAB Environment Step 2: Identifying the variables (Table 28.3)

Step 3: Simulating the model in the MATLAB environment. Results of system model simulation.

Figures 28.1 and 28.2 shows the FMS modeling for the given 4 job 3 machine problem. Once the modeling is created next phase is to check for the structural and behavioral properties of the net structure. As shown in Fig. 28.1 the model of each job should spend some time at each station, in a predetermined order. Figure 28.3 shows the property check whether is net structure modeled satisfy all the condition of the manufacturing system. Net is bounded means there will be no overflows in the buffer. Liveness shows a complete absence of deadlock in the manufacturing system. Once the property of the net is checked, next step is to analyze the system. There

P1	Job 1 in queue
P2	Operation 2 of Job 1 on machine 2
P3	Operation 3 of Job 1 on machine 3
P4	Job 1 finish
P5	Operation 1 of Job 2 on machine 2
P6	Operation 2 of Job 2 on machine 1
P7	Operation 3 of Job 2 on machine 3
P8	Job 2 finish
P9	Operation 1 of Job 3 on machine 3
P10	Operation 2 of Job 3 on machine 2
P11	Operation 3 of Job 3 on machine 1
P12	Job 3 finish
P13	Operation 1 of Job 4 on machine 2
P14	Operation 2 of Job 4 on machine 3
P15	Operation 3 of Job 4 on machine 1
P16	Job 4 finish
P17	Loop operation 1 for Job 1
P18	Loop operation 2 for Job 1
P19	Loop operation 3 for Job 1
P20	Loop operation 1 for Job 2
P21	Loop operation 2 for Job 2
P22	Loop operation 3 for Job 2
P23	Loop operation 1 for Job 3
P24	Loop operation 2 for Job 3
P25	Loop operation 3 for Job 3
P26	Loop operation 1 for Job 4
P27	Loop operation 2 for Job 4
P28	Loop operation 3 for Job 4

Table 28.3 Description oftimed Petri net model inFig. 28.2

are different ways of analyzing the net-like coverability tree, incidence matrix, and simple reduction rules. Here the net is analyzed using matrix equation that governs the dynamic behavior of the concurrent system as shown in Figs. 28.4, 28.5, and 28.6.

Step 4: Optimizing using GA

GA starts with a set of solutions called population. Chromosomes from the population are occupied and used to form a new population. This is motivated by the desire, that the new population will be better than the old one in terms of a fitness criterion. Solutions that are selected to form new solutions (offspring) are selected according to their fitness—the more suitable they are the more chances they must reproduce. There



Fig. 28.1 FMS processing of 4 jobs & 3 M/c



Fig. 28.2 FMS processing of 4 jobs & 3 M/c in MATLAB

MATLAS R2015a	Value	100														/01	1000	* > - A	Carab Day mark	- 0)
NURE PLUIS Arts	VANJICLE	NEA.													_	0		12290	Search Document	ucon 🖌
🔛 🔀 Open 🕶 Rows Columns 🚞	Tanapose																			
New forn 🔒 Pirit 💌 👘 Inset	Celete 📙 Sot 🔻																			
VADAGE SELECTON	Er																			
💠 🕸 🗊 📕 🕨 C. 🕨 Program Files 🕨 MATUA	3 + R2016a + bin +	÷																		
Current Folder	🕑 🗾 Editor - I	badr.m							2	Variable	s - PNTs	et					()	Workspace		
🗋 Name+	315)	PNTo	et X															Name +	Value	
i diregitty	E tul net		-															HW	0	
B registry	-																	d sind	0	
i 🛙 🕄	, in the second se	-				:												RNTglobal	bri stuct	
8 📘 win64	, in the second se			č														RNImodel	1	
deploytool.bat				0				1			0							the Philinet	Isl of	
L locataxeni	0								-									E PNTer	tv1 stuct	
L kcataxed								0		-	0									
L kodata_urt8.eml											-									
 matablee 	0	. 0	0	0	0		0	0	6		0									
E notice					2.0															
E model	Incidenc	e Mate	11X A =	R0 -	A1															
Davel	Column	512	tecost	17																
E menthal																	I			
Direction	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
mesalson	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
mw.mpiesec.bat	0	0	-1	1	0	0	0	0	0	0	0	0	0	9	0	0	0			
🖸 workerbat	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0			
	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0			
	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0			
	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0			
	0	0	D	0	0	0	0	0	0	-1	1	0	0	0	0	0	0			
	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0			
	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0			
Details	v 0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0			
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0			
	1	0	0	-1	1	0	0	-1	1	0	0	-1	1	0	0	-1	0			
Select a file to view details	Column	s 18 t	through	28																
	0	0	0	0	0	0	0	0	0	0	0									
	Command	Nindow															6	0		
		100																		10.00

Fig. 28.3 Structural properties

are different representation schemes for evaluation with the makespan objective. They are operation-based representation, job-based representation, preference-list-based representation, priority rule-based representation. In general, GA uses three steps—selection, crossover, and mutation (4). Selection based on the fitness (makespan in our case) is the source of exploitation, and crossover and mutation help us to promote exploration (Fig. 28.7).

A generation of a GA contains a population of individuals, each of which corresponds to a possible solution in the search space. Everyone in the population is evaluated with a fitness function to produce a value which indicates the goodness of a solution. Selection helps in bringing forward certain members from the population to apply crossover and mutation on the given set of problem. Crossover takes pairs of individuals and uses parts of each to produce new individuals. Random mutations swap parts of an individual to prevent the GA from getting caught in a local minimum.

Step 4.1: Creation of First Generation/Initialization:

🔺 MATURA R2116a																				-	. 5 X
KOLE PLOTS ATTS	VARABLE	VEN											-			8	81	129	0	Search Documentar	ton P
2 70+ tes tem E	- James																				
Serler APet . Inst C	Set -																				
Sector																					
WHAT STOR	Cr I		-	_	-	-	-	-	-	-	-	-	-	-	-	-					
中中回到 I · C · Pogan Ris · MATUR	8 R2016a P bn P	18.0	_		_	_	_	_		The Las			_	_	_	_	-				.,
Difference	- Course	27	- 11 I		-	-	-	-	- 1	uras	s-na	e.						2. 801	these and the second		
L Nine*	2	Phin I	e xl			_												- Nan	e-	Yalue	
e nuegay	<u>∎</u> 101 <u>∞</u>	_																	đ., .	9	
i cjuj	0	0	.0	0	0	0	-1	1	0	0	0	0	0	٥	0	0	0	120	ans Turka	2 Mart	
B ainf4	0	٥	0	0	0	0	0	0	-1	1	0	0	0	٥	3	0	0	H,	Incól	1	
E depletoribet	0	0	. 0	0	0	0	- 0	0	0	-1	1	0	0	٥	0	0	٥	En	int.	bint	
Distant	0	0	0	0	0	0	0	0	0	٥	-1	1	0	٥	0	0	0	En	lar .	bi gut	
C ktirtand	0	٥	0	0	0	0		0	0	٥	0	0	-1	1	0	0	0				
indiziatini [0	٥	0	D	0	0	0	0	0	0	0	0	0	-1	1	0	0				
4 nataboe	0	0	. 0	0	0	0	0	0	0	0	. 0	0	0	0	-1	1	0				
E mouleon	1	0	. 0	-1	1	0	. 0	-1	1	0	0	-1	1	0	0	-1	0				
C nclad																					
Dead	Columns	111	heough	28																	
Energia																					
Devetore	0	0	. 0	0	0	0	0	0	0	0	. 0										
Deputition	0	0	0	0	0	0	0		0	0											
The monet and	0	0	0	0	0	0	0	0	0	0											
Exoler.bet	0	0	0	0	0	0	0	0	0	0											
	0	0	. 0	0	0	0	0	2	0	0											
	0	0	.0	0	0	0	0	5	0	0											
	0	0	0	0	0	0	0	0	0	0								н.			
	0	0	0	0	0	0	0	0	0	0	0										
	0	0	. 0	0	0	0	0		0	0											
	0	0		0	0	0	0	0	0	0	0										
Details	¥ 0	0	.0	0	0	0	0	5	0	0	0										
	0	0		0	0	٥	0		0	0	0										
	٥	٥	0	0	0	0	٥	0	0	0	0										
Select a file to view details	12802238	11 790	8922	15:														L			
	Boundedoe	53																			
	Command W	Indox																			
1.																					
# 0 0 2 B A	8 6	à.	6																	AD (5)	920
	m	r.,	S.																		1020

Fig. 28.4 Incidence matrix part 1

In our problem, we are using the datasets that have been taken from Kumar [6]. This data set has initially four jobs and three machines. The term makespan refers to the cumulative time to complete all the operations on all machines [7]. It is the time taken from scheduling the first job submitted until the completion of the last job. The objective of the problem is to find a valid schedule that yields the minimum makespan. The initial population can be generated randomly. Each machine has randomly 3–4 jobs placed on them. The total running time for each machine is then calculated. Here the job-based representation is focused out of four. For the same, the chromosome would be [1 2 0 3]. The first job to be processed is j_1 is m_1 , m_2 , m_3 , and processing time is 4, 3 2 [1]. Similarly, each is processed. The genetic search process starts with a randomly generated set of chromosomes called the initial population. The size of the population (pop_size) depends on the solution space. The chromosome representation of all the representation is a stated in Table 28.4. The outline GA code is as shown in annexure I. The simple GA structure used to evaluate the system

000 A03 A03 W000 0000 000 000 <td< th=""><th>A MURTUAS ROSTIGA</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>-</th><th>5 X</th></td<>	A MURTUAS ROSTIGA																			-	5 X
Image:	KONE PLOTS ATTS	1.8VRVU	VEN														28	191	9260	Search Documentation	n 2
Image: State in the loss intervent	2 Rose tos E	- Inere																			
Normal Normal Normal Normal Normal Image: Normal Status Image: Normal Status </td <td>See ton Best . hat 0</td> <td>Nen 1 Set +</td> <td></td>	See ton Best . hat 0	Nen 1 Set +																			
Image: Product State State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State Image: Product State	which 2,573																				
Correction: 0 2 Structure: 0 Description: Descrip	💠 🔶 🔄 📔 🕴 C 🕈 Pogram Files 🕈 MATUA	1 R2016a + bin +										_	_	_							• 2
Image:	Carrent Folder	8 Z fator-1	mén							2	Tarici	s-Mix	8					®×	Workpace		9
i i	D Name -	25.7	P.Tx	X						-									Name -	Yolue	
Image: Second	ສີ 📕 ສວິເຊຊລາງ	🗐 tal 👱																	an a	9	
1 0	ii xyay	0	0		5	0	0		1	0	0							^	Ed sind	0	
0 0	8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	0	0	6	0	0	0	0	5	0	0							- 11	Pulgeta	bigud	
Default FINUTUAL ROBBITIES: Ustand FINUTUAL ROBBITIES: Ustand Exactle State Soudedeess Switze Soudedeess Switze Soudedeess Switze Soudedeess Switze Soudedeess Switze Soudedeess Switze Statematisess Switze States is conservative! Switze Switze Switze	E orsionolar	0	0	0	0	0	0	0	0	0	0							- 8	Aint	bint	
Detailed FIGUTAL ROBENTISS: Value Southers Souther Southers Souther The set is bouckets Souther The set is bouckets Souther The set is bouckets Souther The set is conservative! Souther The set is countive! Souther	Dicatami																	- 11	Plar	bigut	
Ustagetaris Societaris Institution The set is bounded: Sentation The set is bounded: Sentation The set is conservatives: Institution The set is conservative: Institution The set is constitute: Institution The set is constitute: Sentation are senting: The set is constitute: Sentation are senting: The set is constituent: Connect Unities The set is constituent: Sentation are senting: The set is constituent: Connect Unities The set is constituent: Sentation are senting: The set ise	Citatand	52800708	11 190	10:13	\$:													- 11			
Willow Considers Small The set is boulde! Small The set is boulde! Small The set is conservative! Small The set is constitue! Small The set is conset is set is the set is constitent! <	Lickourters																	- 11			
In state In sol : 10 docted: In state Conservativess In out in the set is conservative! In set is conservative! In out in the set is conservative! In set is conservative! In out in the set is conservative! In set is conservative! In out in the set is conservative! In set is conservative! In any sector in In set is conservative! In any sector in In set is conservative! In any sector in In set is conservative! In any sector in the set is conservative! In set is conservative! In any sector in the set is conservative! In set is conservative! In any sector in the set is conservative! In set is conservative! In any sector in the set is conservative! In set is conservative! Sector if the set is in any set	Rebuilder	Boundedn	255															- 11			
Sendet Inservertigeness Description Inservertigeness Sectraflex wes stable 0 Description 0 Connact Under 0 Connact Under 0	Sector	The pet	15 202	55652														- 11			
Image: Construction Image: Construction Image: Construction Image: Construction <td>Sneutr</td> <td></td> <td>- 11</td> <td></td> <td></td> <td></td>	Sneutr																	- 11			
In motion In minimum interview Devotopin Interviewing Interviewing Interviewing <tr< td=""><td>] test</td><td>The net</td><td>is con</td><td>sont.</td><td>iw!</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>- 11</td><td></td><td></td><td></td></tr<>] test	The net	is con	sont.	iw!													- 11			
Deskup Impediate Visital report invalues Impediate Premients Impediate Sectaficures dash Impediate Sectaficures dash Impediate Connect Weday Impediate Impediate Impediate Impediate<	Breedat																	- 11			
Manuar Applitiments Designed The set is repetitive! Totalitesy The set is constituent! Minial report invariants 0 Deals * Sectrate 0																		- 11			
Barlend The set is repetitive: traintexy: traintexy: sec:straints: 0 0	E na noextat	Repetiti	PECESS															- 11			
Deals *Lineary the set is consistent! being support invariants 0	E soletat	The net	is rep	etitia	e!													- 1			
Deals *inial report invariants >=2reminents >=2reminents \$=0 = 0 = 0 >=0 =0		Consiste	ocy															- 8			
Minial report instants Dab * sectafic tails 0		The net	is con	sister	:!																
Deals v Advantiants 0		and and																			
Secultation state 0	Dak	V D. Degard	245500		114043													- 11			
Sector fit tares stals 0 <td></td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>8</td> <td>0</td> <td>1</td> <td>0</td> <td></td> <td>0</td> <td>- 11</td> <td></td> <td></td> <td></td>		0	0	0	0	0	0	0	1	0	0	8	0	1	0		0	- 11			
Seed a Returned Stable 0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0				
Seets fit tailed bit 0		0	0	0	0	0	0	0	3	0	0	0	0	1	0	0	0				
0 0	Select a file to view details	0	٥	0	0	0	0	0	1	0	0	0	0	1	٥	0	0				
0 0 </td <td></td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>٥</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td></td> <td></td> <td></td> <td></td>		0	0	0	0	0	٥	0	0	0	0	0	0	0	1	0	0				
		0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	<u>,</u>			
■ ■ 0 0 2 B 2 A 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		Command 1	Rindow															9			
🖬 0 0 8 🖬 🗑 😽 🗑 🚺 🗊 🕹 🛛 0 0 8	I •																				
	a o o e a á	H ()	4	5																AD 40	900 1000

Fig. 28.5 Incidence matrix part 2

is given below. The population at time t is represented by variable s, with the initial population of random estimates as s (0).

```
Procedure GA

t = 0;

initialize s(t);

evaluate s(t);

begin

t = t + 1;

select s(t) from s(t - 1)

reproduce pairs in s(t)

evaluate s(t);
```

Step 4.2: Population evaluation

No. No. No. No. No. No. No. No. Come - Rame Come - Rame Description Description <thdescripti< th=""><th>MATLAS R2016a</th><th>1</th><th></th><th>lucity</th><th></th><th></th><th>~</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>_</th><th></th><th></th><th></th><th></th><th></th><th>20</th><th>-</th><th></th></thdescripti<>	MATLAS R2016a	1		lucity			~											_						20	-	
Image: Description Image: Description Image: Description	KME PLOIS	APR.	,	VAGA	ett		EN				_		_							Ľ	00.	1.949	122	00	earch Documents	son 🕨
het inter inter inter het inter inter <td< th=""><th>2 ZOpen - Ross</th><th>Columns</th><th>H</th><th>Ť.</th><th>Transpos</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></td<>	2 ZOpen - Ross	Columns	H	Ť.	Transpos																					
Note: C Particle Values C Name Values C Name Name <th< td=""><td>lev ton Plat +</td><td></td><td>itset. I</td><td>Delete 1</td><td>Sot 🔻</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></th<>	lev ton Plat +		itset. I	Delete 1	Sot 🔻																					
Image: Control (Control (Contro) (Control (Control (Control (Control (Control (Control (Control (VARIAL 1	111110		ECF.																						
Part Hole © Market Projecti X Market Projecti X Market Projecti X Market Name	···••••	togram Files +	MATLA	8 + R201	léa + bi	•																				
Name: Projekt X Nine: X Indepty State	unent Folder (9	Editor - b	n.tool										Z	Variables	- PNIne									Θx	Workspace	
Stringsby using using thy using septyments Bit Ing 0 1 0	Name -	PNTglo	tal X	PATet	x																				Name+	Whe
ation 0 1 0 <td>- märeoistre</td> <td>E trint</td> <td>-</td> <td></td> <td>_</td> <td></td> <td>Hw</td> <td></td>	- märeoistre	E trint	-		_																				Hw	
1 1 0 1 0	registry	C	_	,	_		_	_	_	_	_	_	_			_	_	_	_	_		_			Hd simil	0
Initial Optigopolari Loticatari Lotadarii L	a uti																								PNToctal	bil struct
Bolycowskit 0 0 1 0 <td< td=""><td>win54</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td></td><td>PNImodel</td><td>1</td></td<>	win54	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0		PNImodel	1
Indust 0 <td>deploytool.bat</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>¢</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td></td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td></td> <td>PNTeet</td> <td>hi or</td>	deploytool.bat	0	0	1	0	0	0	¢	0	0	0	0	0	0	0	0	0		1	0	0	0	0		PNTeet	hi or
Indust 0 <td>lodataumi</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td></td> <td>PNTvar</td> <td>hel struct</td>	lodataumi	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0		PNTvar	hel struct
One of the seconds. O C S	ledataxed	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0			
Autober Invisitati Rectation Rect	lodata utičumi	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	1	0			
Bobilitati mexiat 0	 mafabere 	0	0	0	0	0	0	0	1	0	0	0		0	0	0	0		0	0	0		1			
In crast 0<	ted block			0	0	0	0			6	1	0			0	0				0	0					
Prodit 0 <td>mocbat</td> <td></td> <td>1</td> <td></td> <td></td> <td></td> <td></td>	mocbat																					1				
Predid 0 <td>mexbet</td> <td>,</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>~</td> <td>-</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>2</td> <td></td> <td></td> <td></td> <td></td>	mexbet	,									~	-										2				
Operation O C O	mexpl												-							0						
Decompting 0	In mexentibal	0	0	0	0	0	0	¢	0	0	0	0		0	1	0	0		0	0	0		0			
Developm 0<	meisetupom	0	0	Q	0	0	0	¢	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0			
Improjentati sovietati 1 0	merutisam	0	0	0	0	0	0	¢	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0			
Columns 23 through 28 v Consust Water Consus	mw_mplesecbat	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	D			
Status Window Contract Win		Column	5 23 1	through	28																			*		
rabit v rabit v rabit v rabit v rabit v rabit and s g g statis s g s		Commandia	Volte																					R		
NCTURE Faith initialized in 1.46 seconds. rabit rabit rabit state 9 Stects Net beine oftek atts = f <		700	lbox 7	ath Car	the rea	ad in (1.06 1	conds.																^		
sali v potool sali v ats = Sects Bito derestali Ats = F riggine (Rightor Petri Bet Toolbox - Demo version) with properties: V		NATLAS	B Fath	initia	alized	in 1.4	16 sec	nds.																		
sali v at5 = 5 Select a Bie buier oftali at5 = f Yigure (Maplex Petri Bet Toolbox - Demo Version) with properties: v H		>> 101	fool																							
dabi v at5 =																										
Selects Bie bo view Setals Selects Bie bo view Setals ALS = f: riggure (Resplox Petri Her Toolbox - Demo version) with properties:	etais v																									
Sebec a Bie to view ofta h: ans = /f. Yigure (Moplow Petri Bot Toolbox - Demo Version) with properties:		a																								
Sebeca Be buiew oftab and = // Figure (Maybox Petri Bet Toolbox - Demo version) with properties: N			8																							
Sebela Bielowiew offsh At 3 = A Figure (Maphon Petri Ret Toolbox - teno version) with properties:		1																								
ats = A Figure (Maphow Petri Bot Toolbox - Demo Version) with properties:	Galact a file to view data in																									
ads = / Figure (Rephon Petri Set Toolbox - Demo version) with properties:	According to their octains																									
ft rigure (Mayhow Petri Set Toolbox - Demo version) with properties:		à15 =																								
		ft rig	ire (M	tsebea i	Petri /	Set To:	olbox -	- Demo	versio	n) vit	th prot	ertie	5:												<	
	N.																								1	
		-		-	_		-																	-		

Fig. 28.6 Command window answer

The fitness parameter [fit (c)] considered is the makespan time. The initial population is evaluated for makespan.

Step 4.3: Selection of new population

The process of selecting the chromosomes to represent the next generation has the following steps:

 Conversion of the fitness parameter values to a new fitness value [new_ fit (c)]. A fitness function considered here, which is suitable for minimization objective is

new fit(c) =
$$1 - \text{fit}(c)/F$$
 (28.1)

where fit (c) = makespan time corresponding to chromosome.

F is the sum of the fitness parameter of all chromosomes.





 Table 28.4
 Chromosome representation for four scheme (Reproduced from [1])

	-	-	
S. No.	Representation	Generation	String length
1	Job-based representation	Job to be processed at a particular instant	Number of jobs
2	Operation-based representation	An operation that is context-dependent	Number of operations
3	Priority rule-based representation	Priority dispatching rule	Number of operations/machine
4	Preference-list-based representation	A string of symbol with length of number of jobs	Number of machines

$$F = \sum_{c=1}^{\text{pop_size}} \text{fit}(C)$$
(28.2)

To have more copies of chromosomes with the smallest objective function value, the ratio fit \mathbb{O}/F is subtracted from 1.

2. Conversion of the new fitness parameter to an expected frequency of selection [p(c)].

$$P(c) = \text{new}_{\text{fit}}(c) / \sum_{c=1}^{\text{pop_size}} \text{fit}(c)$$
(28.3)

3. Calculation of the cumulative probability of survival (cp (*c*)). A random selection procedure, which is explained below, generates the next population of the same size. A random number between 0 and 1 is obtained and a chromosome c is selected which satisfies the following condition. This selection process is repeated several times equal to the population size. The method used here is more reliable in that it guarantees that the fittest individuals will be selected and that the actual number of times each is selected will be expected frequency ± 1 . This procedure enables the first chromosome to have multiple copies and the worst will die off [1, 5].

The fitness function calculated here is the output from the Gantt chart. Expected count, new fitness function is given by, 1 - (13/47).

Probability of selection is given by (0.0.7234/2.9999).

The string number here represents the number of job.

Step 4.4: Selection of parents for crossover

Once the makespan is calculated for the different chromosomes, tournament selection is done to filter out those chromosomes which have better makespan values (in this case lesser makespan value) and these chromosomes are then selected to undergo crossover and mutation. In this problem, the tournament size has been taken to be two. Two chromosomes are randomly chosen from the population and their makespan values are compared, whichever chromosome has a lesser makespan value is deemed the winner. After the parents have been chosen, crossover is applied to them.

Chromosomes reproduce among themselves according to a predefined crossover probability. The first child is generated by taking the first portion, from the beginning of the chromosome until the crossover point, of the first parent and the second portion of the second parent, from the crossover point until the end of the chromosome.

Step 4.5: Mutation

Generate the random number, for all chromosomes, if $r \le pm$ mutate the chromosome, where pm is the probability of mutation. Mutation is a diversification strategy

used mostly to avoid the repetition of chromosomes. The mutation followed in this paper is the order-based mutation (OBM). In this, we pick two loci in the chromosome at random and exchange their genes. For example, 3 0 1 2 is mutated as 3 2 1 0.

Step 4.6: Termination

The above process will be repeated for a fixed number of generations.

The working of the genetic algorithm is explained in Step 4. The GA parameters considered are given in Table 28.5. The parameters used in GA are

Initial population, corresponding fitness values (population evaluation), selection of chromosomes for the next generation, and chromosomes in the mating pool for the considered example are given in Table 28.6. The parents selected for crossover from the mating pool and corresponding offsprings after crossover, chromosomes selected for mutation, new population after mutation, and fitness values for the considered example are given in Table 28.7.

Parameters	Particulars	Remarks									
Population size	A fixed number of individual form the GA population	10									
Crossover rate	The probability for an individual to perform crossover	0.8									
Mutation rate	The probability for an individual to perform mutation	0.05									

Table 28.5 Parameters used in GA

String No	Initial position	Fitness value	Expected count	Probability of selection	Cumulative probability	Random No	String No.	Mating pool
0	1203	13	0.7234	0.2411	0.2412	0.22	2	3021
1	2130	9	0.8085	0.2695	0.5107	0.02	1	2130
2	3021	14	0.7021	0.234	0.7447	0.36	3	0312
3	0312	11	0.7659	0.2553	1	0.22	2	3021
		47	2.9999					

Table 28.6 Evaluation and reproduction phase

Table 28.7 Population cross over

Mating pool	Random No.	Selected over crossover	Population after crossover	Random mutation	Population after mutation	Fitness value
3021	0.8900	Yes	3012	0.5000	3210	16
2130	0.7300	No	2130	0.3214	2130	9
0312	0.6200	No	0321	0.2200	0321	11
3021	0.8900	No	3021	0.0300	3021	14

The best makespan and corresponding schedule are given below for the example problem for the job-based representation schemes considered in this example. The best makespan obtained is 9 and Best schedule: 2-1-3-0.

28.4 Conclusions

The scheduling is one of the important aspects of the smooth functioning of flexible manufacturing system. In the present work combined approach of Petrinet and genetic algorithm is used to produce near to optimal result. Here only one performance parameter is considered that is makespan. Due to good modeling power, Petri net is used as this feature is required to model a real manufacturing system. Genetic algorithms are powerful optimization tool and gives better result for combinatorial problems. Out of the four representations scheme the job-based representation is addressed for the optimum makespan. The hybrid approach plays a vital role in scheduling the system and optimizing the makespan.

In the future, the focus must be on a more complex manufacturing system with different representation schemes and with minimum number of constraints. Also, the focus on multi-objective optimization with conflicting objectives/performance measures will be addressed.

Annexure

```
Population population = new Population();
Individual fittest;
Individual secondFittest;
int generationCount = 0;
public static void main(String[] args) {
    Random rn = new Random();
    GA demo = new GA();
    //Initialize population
    demo.population.initializePopulation(10);
    //Calculate fitness of each individual
    demo.population.calculateFitness();
    System.out.println("Generation: "
+ demo.generationCount + " Fittest: "
+ demo.population.fittest);
```

 $//\ensuremath{\mathsf{While}}$ population gets an individual with maximum fitness

```
while (demo.population.fittest < 5) {
     ++demo.generationCount;
            //Do selection
            demo.selection();
            //Do crossover
            demo.crossover();
            //Do mutation under a random probability
            if (rn.nextInt()\%7 < 5) {
                demo.mutation();
            }
            //Add fittest offspring to population
            demo.addFittestOffspring();
            //Calculate new fitness value
            demo.population.calculateFitness();
            System.out.println("Generation: "
        demo.generationCount
                                  +
                                           " Fittest: "
                                                              +
demo.population.fittest);
       }
        System.out.println("\nSolution found in generation " +
demo.generationCount);
       System.out.println("Fitness: "
+ demo.population.getFittest().fitness);
        System.out.print("Genes: ");
        for (int i = 0; i < 5; i++) {
System.out.print(demo.population.getFittest().genes[i]);
       }
       System.out.println("");
    }
    //Selection
    void selection() {
        //Select the most fittest individual
        fittest = population.getFittest();
        //Select the second most fittest individual
        secondFittest = population.getSecondFittest();
    }
    //Crossover
    void crossover() {
       Random rn = new Random();
        //Select a random crossover point
        int crossOverPoint = rn.nextInt
```

```
(population.individuals[0].geneLength);
        //Swap values among parents
        for (int i = 0; i < crossOverPoint; i++) {</pre>
            int temp = fittest.genes[i];
            fittest.genes[i] = secondFittest.genes[i];
            secondFittest.genes[i] = temp;
        }
    }
    //Mutation
    void mutation() {
        Random rn = new Random();
        //Select a random mutation point
        int mutationPoint
= rn.nextInt(population.individuals[0].geneLength);
        //Flip values at the mutation point
        if (fittest.genes[mutationPoint] == 0) {
            fittest.genes[mutationPoint] = 1;
        } else {
            fittest.genes[mutationPoint] = 0;
        }
        mutationPoint
= rn.nextInt(population.individuals[0].geneLength);
        if (secondFittest.genes[mutationPoint] == 0)
{
            secondFittest.genes[mutationPoint] = 1;
        } else {
            secondFittest.genes[mutationPoint] = 0;
        }
    }
    //Get fittest offspring
    Individual getFittestOffspring() {
        if (fittest.fitness > secondFittest.fitness) {
            return fittest;
        }
       return secondFittest;
    }
    //Replace least fittest individual from most fittest
offspring
    void addFittestOffspring() {
        //Update fitness values of offspring
        fittest.calcFitness();
        secondFittest.calcFitness();
```

```
//Get index of least fit individual
        int leastFittest
Index = population.getLeastFittestIndex();
        //Replace least fittest individual from most fittest
offspring
       population.individuals[leastFittestIndex]
                                                               =
getFittestOffspring();
  }
}
//Individual class
class Individual {
    int fitness = 0;
    int[] genes = new int[5];
    int geneLength = 5;
    public Individual() {
        Random rn = new Random();
        //Set genes randomly for each individual
        for (int i = 0; i < genes.length; i++) {
            genes[i] = Math.abs(rn.nextInt() % 2);
        l
        fitness = 0;
    }
    //Calculate fitness
    public void calcFitness() {
        fitness = 0;
        for (int i = 0; i < 5; i++) {
            if (genes[i] == 1) {
       ++fitness;
            }
       }
    }
}
//Population class
class Population {
    int popSize = 10;
    Individual[] individuals = new Individual[10];
    int fittest = 0;
    //Initialize population
    public void initializePopulation(int size) {
        for (int i = 0; i < individuals.length; i++) {</pre>
            individuals[i] = new Individual();
        }
```

```
}
    //Get the fittest individual
    public Individual getFittest() {
        int maxFit = Integer.MIN_VALUE;
        int maxFitIndex = 0;
        for (int i = 0; i < individuals.length; i++) {</pre>
            if (maxFit <= individuals[i].fitness) {</pre>
                maxFit = individuals[i].fitness;
                maxFitIndex = i;
            }
        3
        fittest = individuals[maxFitIndex].fitness;
        return individuals[maxFitIndex];
    3
    //Get the second most fittest individual
    public Individual getSecondFittest() {
        int maxFit1 = 0;
        int maxFit2 = 0;
        for (int i = 0; i < individuals.length; i++)</pre>
{
            if (individuals[i].fitness > individuals[maxFit1].
fitness) {
                maxFit2 = maxFit1;
                maxFit1 = i;
            } else if (individuals[i].fitness
                                                                >
individuals[maxFit2].fitness) {
                maxFit2 = i;
            }
        }
        return individuals[maxFit2];
    3
    //Get index of least fittest individual
    public int getLeastFittestIndex() {
        int minFitVal = Integer.MAX_VALUE;
        int minFitIndex = 0;
        for (int i = 0; i < individuals.length; i++) {</pre>
            if (minFitVal >= individuals[i].fitness) {
                minFitVal = individuals[i].fitness;
                minFitIndex = i;
            }
        }
       return minFitIndex;
    }
    //Calculate fitness of each individual
    public void calculateFitness() {
        for (int i = 0; i < individuals.length; i++) {</pre>
            individuals[i].calcFitness();
        }
        getFittest();
    }
}
```

References

- 1. Ponnambalam, S.G, Aravindan, P., Sreenivasa Rao, P.: Comparative Evaluation of Genetic Algorithms for Job-Shop Scheduling, Vol. 12(6), pp. 560–574. Taylor & Francis (2010)
- 2. Murata, T.: Petri nets: properties, analysis, and application. IEEE 77(4), 541-580 (1989)
- Zurawski, R., Zhou, M.: Petri nets and industrial applications: a tutorial. IEEE. Trans. Ind. Electron. 41(6), 567–583 (1994)
- Filho, M.G., Barco, C.F., Neto, R.F.T.: Using genetic algorithms to solve scheduling problems on flexible manufacturing systems (FMS): a literature survey, classification and analysis. Flex. Serv. Manuf. J. 26, 408–431 (2014)
- Muni Babu, P., Himasekhar Sai, B.V., Sreenivasulu Reddy, A.: Optimization of make-span and total tardiness for flow shop scheduling using genetic algorithm. Int. J. Eng. Res. Gen. Sci. 3(3) (2015)
- 6. Kumar, O.C.: Scheduling of flexible manufacturing system using timed Petri nets. In: International Conference on Automation, Indore (1995)
- 7. Vadher, J., Patel, M.B.: Dynamic scheduling of manufacturing system with stochastic timed petrinet: a genetic algorithm approach. In: Proceedings of ISECON, Vol. 25 (2008)