Modeling and Optimization of Part Type Selection and Loading Problem in Flexible Manufacturing System Using Real Coded Genetic Algorithms

Wayan F. Mahmudy, Romeo M. Marian, Lee H. S. Luong

Abstract—This paper deals with modeling and optimization of two NP-hard problems in production planning of flexible manufacturing system (FMS), part type selection problem and loading problem. The part type selection problem and the loading problem are strongly related and heavily influence the system's efficiency and productivity. These problems have been modeled and solved simultaneously by using real coded genetic algorithms (RCGA) which uses an array of real numbers as chromosome representation. The novel proposed chromosome representation produces only feasible solutions which minimize a computational time needed by GA to push its population toward feasible search space or repair infeasible chromosomes. The proposed RCGA improves the FMS performance by considering two objectives, maximizing system throughput and maintaining the balance of the system (minimizing system unbalance). The resulted objective values are compared to the optimum values produced by branch-and-bound method. The experiments show that the proposed RCGA could reach near optimum solutions in a reasonable amount of time.

Keywords—Flexible manufacturing system, production planning, part type selection problem, loading problem, real-coded genetic algorithm.

I. INTRODUCTION

RAPID market changes (changing customer needs), peaks in demand for product quantity (e.g. new gadgets – tablet PC, mobile phones, etc.), concerns for product quality, and requirements to dramatically increase product mix have forced manufacturing industries to enhance their flexibility. Flexible manufacturing system (FMS) is designed to cope with these conditions by using high technologies and automation in transfer lines which enable factories to reconfigure rapidly to produce a variety of products by using same resources [1]-[3]. Due to the high investment required, higher resources utilization must be achieved and this issue can be resolved by establishing a good production planning. This planning will also increase productivity by maximizing system throughput and enable early return on investment.

The planning problem in FMS is related with the arrangement of parts and technological equipments such as tools, fixtures and pallets, and the determination of the type and quantity of the products which are made before starting production [4], [5]. The planning problems can be divided into two sub problems, an aggregate production planning and a short term planning (or a production planning) [6]. The aggregate production planning produces a master schedule containing part mix, production rates and lot sizes. The production planning gives interface between aggregate production planning and daily operation of the FMS. There are several issues in the production planning stage such as part type selection problem, machine grouping problem, production ratio problem, resource allocation problem, and loading problem [7], [8].

Depend on the specific characteristic of manufacturing environments, various combination of some production planning problems have been considered in the literatures. For example, Bilgin & Azizoglu [9], Chan & Swarnkar [10], and Chen & Ho [11] solved the machine loading problem. Swarnkar & Tiwari [12], Choudhary, Tiwari & Harding [13], Biswas & Mahapatra [14], Ponnambalam & Kiat [15], and Prakash et al. [16] solved the part type selection and loading problem simultaneously. Tabucanon, Batanov & Basu [17] solved the part type selection and loading problem simultaneously in the first stage and used the result on this stage to determine the production ratio in the next stage. However, the routing flexibility was not considered. Kim et al. [18] solved the loading problem and the partial machine grouping while considering tool life constraints. Seok Shin, Park & Keun Kim [19] solved the loading problem while considering a various flexibility such as machine, sequence, tool and process routing. This paper focuses on the part type selection and loading problem with machine and tool flexibility.

The part type selection problem and the loading problem are parts of the production planning problems which are strongly related and heavily influence the system's efficiency [14], [20]. The part type selection problem deals with selection of set of part types (products) which must be produced immediately from a number of part types as there are different due dates, limited number of machines, limited tool magazines capacity of each machine and limited number of tools. The loading problem is concerned with allocation of operations for selected part types and loading appropriate tools to the machines [7], [21]. Solving part type selection and machine loading problem simultaneously will produce a better solution

W. F. Mahmudy is a PhD student at School of Advanced Manufacturing & Mechanical Engineering, University of South Australia and a lecturer at Department of Computer Science, Brawijaya University (UB), Indonesia (e-mail: wayan_firdaus.mahmudy@mymail.unisa.edu.au, wayanfm@ub.ac.id).

R. M. Marian, PhD is Senior Lecturer and Program Director at School of Advanced Manufacturing & Mechanical Engineering, University of South Australia, Australia (e-mail: romeo.marian@unisa.edu.au).

L. H. S. Luong is a professor at School of Advanced Manufacturing & Mechanical Engineering, University of South Australia, Australia (e-mail: lee.luong@unisa.edu.au).

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

as higher throughput of the FMS is achieved while keeping the balance of machines' workload [3]. Here, the efficient allocation of production resources will be achieved.

Solving the part type selection problem and the machine loading problem simultaneously require a good approach to achieve a good result on reasonable amount of time. Genetic algorithms (GAs) are regarded as the powerful method to solve a complex problem with a large search space [22]. GAs have an ability to escape from local optima as they can jump randomly from one sequence to another sequence [23]. The power of GAs to solve various complex problems has attracted a lot of researchers to do research in this area. However, a simple GAs is insufficient to solve any complex problems in engineering. A proper representation, developing appropriate search operators and hybridizing it with other methods are an important key for its successful implementation and becoming challenging tasks [24], [25]. In addition, a good strategy to avoid premature convergence which produces local optimum solution should be developed [26]. This paper as an extension of [27], [28] attempts to develop a new representation using an array of real numbers which could produce good solutions efficiently by using simple genetic operators. The novel proposed chromosome representation is designed to produce only feasible solutions which minimize a computational time needed by GAs to push its population toward feasible search space or repair infeasible chromosomes.

II. LITERATURE REVIEW

The part type selection problem and the loading problem in the FMS environments have received significant attentions from researchers who proposed a various approaches to solve the optimization of the production planning problems such as mathematical programming [17], Lagrangean relaxation approach [9], genetic algorithms [11], [13], particle swarm optimization [14], [15], ant colony optimization [10], immune algorithm [16], two-stage heuristics based on a bin-packing algorithms and a simple search technique [18], multi-agent system [20], and symbiotic evolutionary algorithm [19]. A combination of two methods was also used such as hybridizing genetic algorithm with simulated annealing [21], [29], and hybrid tabu search and simulated annealing-based [12]. Here, heuristic methods is widely used since direct methods which are based on mathematical programming and smart enumeration are not practical to solve these complex problems [22].

The extensive works were shown in several literatures. For example, Tabucanon, Batanov & Basu [17] used simulation to evaluate the solution of part type selection and loading problem produced by batching approach. They formulated mathematical programming method to maximize the number of part types in each batch. Denizell & Sayin [30] developed a mathematical programming model to solve the part type selection problem that considering due dates. Choudhary, Tiwari & Harding [13] addressed the problems by using a GAs with chromosome differentiation. A sexual differentiation of the chromosomes was applied to maintain a diversity of the population and explore the search space

extensively. Tiwari et al. [21] proposed a constraints-based fast simulated annealing algorithm to solve a combination of part-type selection and operation allocation on machines. Their proposed algorithm which was performed by a combination of a GA and a simulated annealing (SA) had a capability to escape from local optimum and provide good solutions. Biswas & Mahapatra [14] proposed modified particle swarm optimization (PSO) to solve machine loading problems. This algorithm attempted to maintain the balance of the system while regarding the occurrence of technological constraints such as the availability of machining time and tool slots. Ponnambalam & Kiat [15] also used PSO to solve machine loading problems. This algorithm is equipped with two local search method to improve the solution quality. They applied two objectives, minimizing system unbalance and maximizing system throughput, while satisfying the technological constraints. Tiwari, Kumar Jha & Bardhan Anand [20] developed a combinatorial auction-based heuristic for multi-agent system to solve the problems. This approach was used to deal with a huge search space of the part type selection and machine loading problems. Even though all these researchers reported promising results, several simplicities were made to reduce the complexity of the problems.

Part type selection and machine loading are NP-hard problems [14]. The complexity of the problems is harder when flexibilities of operations are considered. For example, each part has alternative routes (routing flexibility) which refer to a possibility of operation is processed on alternative machines with alternative tools. For simplicity, Tabucanon, Batanov & Basu [17] did not consider the routing flexibility. Denizell & Sayin [30] and Pacciarelli [31] considered FMS that consists of all general-purpose machines where the functionality of a machine is only determined by the set of tools loaded in their tool magazine. Here, each part is only processed by one machine. Furthermore, Swarnkar & Tiwari [12], Choudhary, Tiwari & Harding [13], Biswas & Mahapatra [14], Ponnambalam & Kiat [15], Prakash et al. [16] and Tiwari, Kumar Jha & Bardhan Anand [20] did not mention specific tool types and its availability; they only mentioned the number of slots needed by the tools. In contrast, this paper considers machine and tool flexibility and also limited numbers of tool types.

III. PROBLEM DESCRIPTION

This paper considers a FMS which consists of several computers numerically controlled (CNC) machines equipped with a tool magazine which has limited tool slot capacity. The machines can perform different operations when they are equipped with different tools. A limited number of tools are available and each tool requires a number of slots when it is assigned to a machine. When several jobs (part types) arrive, the system must select a set of part types which must be produced immediately as there is a limitation of machines and its tool slot capacity and tools availability. This approach is considered as batching approach as all part type should be grouped into several production batches [17].

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

Each part type has a production requirement in form of sequence of operations. Each operation can be processed on several alternative machines with several alternative tools. This paper also considers unrelated machines approach where time needed for parts' operations depend on the assigned machine.

A. Subscripts

$p = 1,\dots,P$ part type
$o = 1,\dots,O_p$ operation of part type p
$t = 1,\dots,T$ tool type
$m = 1,\dots,M$ machine type

B. Parameters

$MS_m =$ tool slot capacity of machine m
$TN_t =$ number of tools type t
$TS_t =$ number of slots required by tool type t
$PS_p =$ batch size of part type p
$PV_p =$ value (price) of part type p
$MOP_{po} =$ set of possible machines on which operation o of part type p can be performed
$TM_{pomt} = \{1,0\}$: 1 if tool t is required for processing operation o of part type p on machine m
$T_{pom} =$ processing time of operation o of part type p on machine m

C. Decision variables

$X_p = \{1,0\}$:   1 if part type p is selected in the batch
$X_{pom} = \{1,0\}$: 1 if machine m is selected to process operation o of part type p
$Y_{mt} = \{1,0\}$:   1 if tool t is loaded to the machine m

D. Objectives

A various objectives had been considered in the references such as maximizing system throughput [3], [12], [13], [15]-[17], [21], [23], [29], [30], [32], maintaining the balance of the system [3], [12]-[16], [19], [21], [23], [29], [32], minimizing part movement [19], minimizing tool changeovers [19] and minimizing production cost [33]. However, most of references considered two common objectives, maximizing system throughput and maintaining the balance of the system (minimizing system unbalance). System throughput and system unbalance can be calculated in different ways as follows:

1) Maximizing System Throughput

$$\text{Maximize} : \sum_{p=1}^{P} X_p \tag{1}$$

$$\text{Maximize} : \sum_{p=1}^{P} X_p PS_p PV_p \tag{2}$$

Maximizing system throughput can be achieved by maximizing the number of selected part types in each batch which can be expressed as in (1). This objective means minimizing a time lost for tools changeover. Tabucanon, Batanov & Basu [17], Kumar & Shanker [3] used this objective function. Another way to maximize system throughput is by maximizing the value (price or profit) or the sum of batch size (if all part types have equal price or profit) of selected part types in each batch as shown in (2). Kumar & Shanker [3], Choudhary, Tiwari & Harding [13], Prakash et al. [16], Ponnambalam & Kiat [15], and Yogeswaran, Ponnambalam & Tiwari [29] used this objective function.

2) Maintaining the Balance of the System

$$\text{Minimize} : \sum_{m=1}^{M} \left| W_m - \overline{W} \right| \tag{3}$$

where $W_m = \sum_{p=1}^{P} \sum_{o=1}^{O_p} X_{pom} T_{pom}$

and $\overline{W} = \left(\sum_{m=1}^{M} W_m \right) / M$

$$\text{Maximize} : \max_{m=1,\dots,M} \left\{ \sum_{p=1}^{P} \sum_{o=1}^{O_p} X_{pom} T_{pom} \right\} \tag{4}$$

$$\text{Minimize} : \sum_{m=1}^{M} \left| SP_m - W_m \right| \tag{5}$$

$$\text{Minimize} : \sum_{m=1}^{M} \left(SP_m - W_m \right) \tag{6}$$

Maintaining the balance of the system can be achieved by minimizing system unbalance as expressed in (3) where W_m is workload of machine m and $\overline{W}$ is the average machine workload. Seok Shin, Park & Keun Kim [19] used this objective function. Another way to minimize system unbalance is by minimizing a maximum machine's load as in (4). If length of scheduling period for each machine (SP_m) is determined in advance and overloading of the machines is allowed, the system unbalance may be expressed as in (5). Mukhopadhyay, Midha & Krishna [32], Biswas & Mahapatra [14] used this objective function. However, if overloading of the machines is not allowed, (6) is used. Here, length of scheduling period becomes a maximum machine workload. Choudhary, Tiwari & Harding [13], Biswas & & Mahapatra [14] used this objective function.

E. Constraints

While minimizing system unbalance and maximizing system throughput, several technological constraints must be satisfied as follows:

$$\sum_{o=1}^{O_p} \sum_{m=1}^{M} X_{pom} = O_p X_p \quad p = 1,\dots, P \tag{7}$$

$$\sum_{m \in MOP_{po}} X_{pom} = X_p \quad p = 1,\dots, P \quad o = 1,\dots, O_p \tag{8}$$

$$Y_{mt} = X_{pom} TM_{pomt}$$
$$p = 1,\dots, P \quad o = 1,\dots, O_p \quad m = 1,\dots, M \quad t = 1,\dots, T \tag{9}$$

$$\sum_{m=1}^{M} Y_{mt} \leq TN_t = O_p X_p \quad t = 1,\dots, T \tag{10}$$

$$\sum_{t=1}^{T} Y_{mt} TS_t \leq MS_m = O_p X_p \quad m = 1,\dots, M \tag{11}$$

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

$$\sum_{p=1}^{P} \sum_{o=1}^{O_p} X_{pom} T_{pom} < SP_m \quad m = 1,..., M \qquad (12)$$

Constraint (7) guarantees that if a part type is selected, all its operations must be performed. This constraint states that operation assignments are equal to the total operations required. Constraint (8) states that each operation of selected part types must be completed on one machine. As there is possibility of operation can be processed on alternative machines, the machine must be determined and the operation must be processed by the chosen machine. Constraint (9) guarantees that if a machine is selected to process an operation of a part type, all the tools needed must be loaded to the machine. Constraint (10) ensures that the number of tools loaded to the machines must not exceed its availability. Constraint (11) ensures that the number of tool slots used on a machine must not exceed the machine's tool slot capacity. Constraint (12) is only used if length of scheduling period for each machine (SP_m) is determined in advance and overloading of the machines is not permitted.

IV. Modeling Using GA

GAs are general purpose search algorithms which imitate a natural evolution process. Candidate solutions are represented by chromosomes which evolve over time (generations) through reproduction and stochastic selection. Along generations these chromosomes become better (with higher fitness value) and at the final generation the best chromosome can be decoded as a near optimum solution [34]. This section describes how real-coded genetic algorithm is used to solve part type selection and loading problem.

A. Chromosome Representation

A suitable chromosome representation determines the successful implementation of genetic algorithms [35]. This paper uses real number representation so GAs which uses this representation can be called real-coded GAs (RCGA). A chromosome is a vector of real number whose size is same with the number of part types. This representation usually was used to solve optimization problems on continuous domains [36]. However, a simple implementation of its operators (crossover and mutation) and possibility to decode one real number into several values (part type's index and its several operations) become the main reason to use this representation for solving part type selection and loading problem.

The construction of a chromosome in our RCGA is shown in Table I. Each element of the chromosome $X=(x_1, x_2, \ldots, x_p)$ corresponds to the continuous position values for p number of part-types. The value of x_i is maintained between 0 and $2^{opMax \times bitMac + bitPart}$. $opMax$ is maximum number of operations of each part type. $bitMac$ is number of bits required to represent a binary number which has largest value of maximum number of alternative machines of each operation. For example, the maximum number of alternative machines of each operation is 5. Therefore, the minimum bits required to represent a binary number between 0 and 5 is 3. $bitPart$ is

number of bits required to represent a binary number which has largest value of number of part types. x_i is stored and treated as a real number when genetic operators (crossover and mutation) are applied. However, x_i will be converted (rounded) to a nearest integer value when decoding operation is performed.

TABLE I
CHROMOSOME CONSTRUCTION

part type	1	2	3	4	5	6	7	8
x	2772	7779	5129	7981	6215	977	9969	1654
part type sequence	6	8	1	3	5	2	4	7
sorted x	977	1654	2772	5129	6215	7779	7981	9969

A smallest position value (SPV) rule is used to get part types sequence. By sorting x in ascending order, we obtain the sequence of part types that are selected for the current batch. To determine which machines are used to process part types' operations, each element of X is decoded into a binary number. Suppose part type 8 has 3 operations, operation 1 can be processed on machines 2 or 3 or 5, operation 2 can be processed on machine 1 or 2 or 6 or 7, and operation 3 can be processed on machine 3 or 4. To choose machines used for processing of operations of part type 8, $x_2=1654$ is converted into a binary number $(11001110110)_2$. Suppose $bitMac$ is 3. For the first operation, we use 3 bits at the right side $(110)_2$ which is equal to 6. As there are 3 possible machines ($n=3$), we apply the following formula:

$$machine\ index = 6\ mod\ n + 1 = 6\ mod\ 3 + 1 = 1$$

mod is modulus operator which gives the remainder of a division. Therefore, the first operation of part type 8 is performed on the first possible machine that is machine 2. By using the next 3 bits and applying the same rule, we obtain that part type 8 is sequentially processed on machines 2, 6 and 4.

After determining the machines for operations, required tools are assigned to the machines. At this step, all the constraints such as availability of tools and empty slots on the machines are checked. For example, after choosing part types 6, 8, 1 and 3 according to the part type sequence as shown in Table I, adding part type 5 to the solution violate the constraints. Therefore, the chromosome states that only part types 6, 8, 1 and 3 are selected for the current batch and the objective functions of the problem are calculated based on these selected part types.

Note that the proposed representation produces only feasible solutions and guarantees that only required tools assigned to the machines. This effort will minimizes a computational time which usually needed by GAs to push its population toward a feasible search space or repeatedly repair infeasible chromosomes [37].

B. Fitness Function

The objective functions of the optimization problem must be converted to a fitness function which is used to measure the goodness of the solution. For example, (2) is used to measure

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

system throughput, (2) should be converted into (13) to produce value between 0 and 1.

$$f_1 = \left(\sum_{p=1}^{P} X_p PS_p PV_p \right) \Big/ \left(\sum_{p=1}^{P} PS_p PV_p \right) \tag{13}$$

Furthermore, (5) is used to measure system unbalance. Here, length of scheduling period for each machine (SP_m) is determined in advance and overloading of the machines is permitted. Minimizing (5) can be converted as maximizing (14) as follow:

$$f_2 = 1 - \left(\sum_{m=1}^{M} |SP_m - W_m| \right) \Big/ \sum_{m=1}^{M} SP_m \tag{14}$$

Finally, the fitness function can be formulated as follow:

$$\text{Maximize} : F = \alpha_1 f_1 + \alpha_2 f \tag{15}$$
$$\alpha_1, \alpha_1 : \quad \text{weigthed parameters}$$

C. Initialization of Population

pop_size of chromosomes are created as an initial population. Here, x_i is generated randomly within its range.

D. Reproduction

On every generation, new chromosomes (*offspring*) are produced by using crossover operator and mutation operator. The number of new chromosomes produced is determined by *crossover-rate* (*cr*) and *mutation-rate* (*mr*) parameters. For example, if population size is *pop_size* then there are *pop_size*×*cr* offspring produced by crossover operator and *pop_size*×*mr* offspring produced by mutation operator for each generation. All offspring produced in this stage are placed in offspring pool. Parents for these reproduction operations are randomly and uniformly chosen from the population.

This paper uses two crossover methods, *flat-crossover* [38] and *extended-intermediate-crossover* [39]. Let $P_1=(p_1^1,\ldots,p_n^1)$ and $P_2=(p_1^2,\ldots,p_n^2)$ are two selected chromosomes as parents for crossover. *Flat-crossover* produces offspring $O=(o_1,\ldots,o_n)$ by generating a random number o_i on interval $[p_i^1,\ldots,p_i^2]$. *Extended-intermediate-crossover* uses a formula $o_i=p_i^1+\alpha_i(p_i^2-p_i^1)$, where α_i is randomly generated on interval $[-0.25, 1.25]$. These crossover methods are randomly chosen in each generation.

Random exchange mutation which is usually applied in permutation representation is used. This mutation works by selecting two genes randomly and exchanging their positions. We also develop mutation method for real number representation, *simple-random-mutation*. If $P=(p_1,\ldots,p_n)$ is selected parent for mutation then offspring $O=(o_1,\ldots,o_n)$ is produced by applying a formula $o_i=p_i+\alpha_i$, where α_i is randomly generated on interval $[-0.1, 0.1]$. These mutation methods are randomly chosen in each generation.

E. Selection

Selection procedure is used to select *pop_size* chromosomes from current population (parents) and offspring pool to perform the next generation. Four common selection methods will be examined to determine which method is most suitable for the RCGA. These selection methods are:

1) Roulette Wheel Selection

Each chromosome from current population (parents) and offspring pool has probability to be selected according to its fitness value. Here, a cumulative probability is calculated and a random number is generated to select the chromosome.

2) Binary Tournament Selection

One chromosome from current population and one chromosome from offspring pool are randomly chosen and compared. The best one will be selected.

3) Elitist Selection

All chromosomes from current population (parents) and offspring pool are placed in one pool and sorted according their fitness value. *pop_size* best chromosomes are selected.

4) Replacement Selection

Each chromosome in offspring pool will be selected to replace its parent if it has a better fitness value than its parent. If the child is produced by crossover operator (by using two parents) then the child will replaces the worst parent.

Note that the binary tournament, elitist and replacement selection guarantee that the best chromosome is always passed to the next generation.

F. Overall RCGA Cycle

The overall RCGA cycle is shown as follow:

Step 0. Setting GA parameters

Parameters: population size *pop_size*, crossover rate *cr*, mutation rate *mr*, maximum number of generations *max_gen*, weighted parameters (α_1 and α_2) for fitness function.

Step 1. Initialization

Let generation *gen*=0.

Generate *pop_size* of random chromosomes.

Step 2. Reproduction

Produce *pop_size*×*cr* offspring by using crossover operator and *pop_size*×*mr* offspring by using mutation operator.

Step 3. Selection

Select *pop_size* chromosomes from population and offspring pool for the next generation.

Step 4. Let *gen* ← *gen* + 1.

If gen=*max_gen* go to Step 2, else Stop.

V. NUMERICAL EXAMPLE

A simple problem set is given to demonstrate how the proposed RCGA solves the problem. There are 3 different machines which have tool slot capacity of 20, 15 and 20 respectively. Length of scheduling period for each machine (SP_m) is 4000 and overloading of the machines is allowed. Furthermore, as shown in Table II, there are 10 tool types where each tool type requires several tool slots on machines' magazine.

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

TABLE II
TOOL TYPES' AVAILABILITY

tool type	1	2	3	4	5	6	7	8	9	10
availability	2	2	2	2	2	3	3	3	3	3
number of slot needed	3	3	4	4	5	5	4	4	3	3

Eight part types are ready to be produced as shown in Table III. Each part type has specific production requirements. For example, part type 1 has 3 operations. Operation 2 can be processed on machine 1 or 3. Machine 1 needs 20 unit times for processing and tools 4, 5 are required. Note that the problem has machine and tool flexibility.

TABLE III
PART TYPE PRODUCTION REQUIREMENT

part type	batch size	value ($)	op	mac	time (seconds)	tools		
1	30	5	1	1	30	1	2	3
			2	1	20	4	5	
				3	20	2	3	
			3	3	30	3	4	
2	30	3	1	1	40	1	2	
			2	2	20	3	4	
			3	3	30	5	6	7
3	30	2	1	1	30	6	7	8
				3	40	8	9	10
			2	2	40	1	10	
				3	30	1	10	
			3	1	20	1	2	
4	30	1	1	3	30	9	10	
				2	20	9	10	
			2	2	30	6	7	
				1	40	6	7	
			3	1	30	3	4	
5	40	4	1	1	40	1	2	3
			2	2	40	4	5	
				3	40	4	5	
6	40	3	1	3	20	7	8	
			2	2	50	9	10	
			3	1	10	3		
7	40	2	1	2	20	3	4	
			2	2	30	1	2	
				3	40	8	9	
8	40	5	1	1	50	1	2	3
				3	40	8	9	10
			2	2	30	4	5	

op: operation, mac: machine, ntool: number of tools required

As shown in Table III, maximum number of operations of each part type is 3 (*opMax*=3) and maximum number of alternative machines is 2 that requires 2 bits for a binary number (*bitMac*=2). Number of part types is eight that requires 4 bits for a binary number (*bitPart*=4). Therefore, the value of each element of chromosome is maintained between 0 and $2^{3 \times 2 + 4}$.

Several GAs parameters must be determined in advance as follows: population size is 100, crossover rate is 0.3, mutation rate is 0.1, maximum number of generations is 500 and weighted parameters (α_1 and α_2) are set equal to 1. Crossover rate of 0.3 means that 100x0.3=30 offspring are produced by crossover operator for each generation. Similarly, mutation rate of 0.1 means that 100x0.1=10 offspring are produced by mutation operator for each generation. Replacement selection is used.

By using these parameters, the RCGA produces an optimum solution for the part type selection and loading problem in less than 1 second. The optimum solution is achieved after 23 generations. This optimum solution is checked by using branch-and-bound method. The best chromosome $X=(793,603,439,1022,344,713,86,426)$ is converted to determine selected part types and its chosen machines for operations as shown in Table IV. Machines' workload and tools assigned are shown in Table V. The increase of the best and average fitness value is depicted in Fig. 1 which shows a fast convergence of the RCGA to optimality.
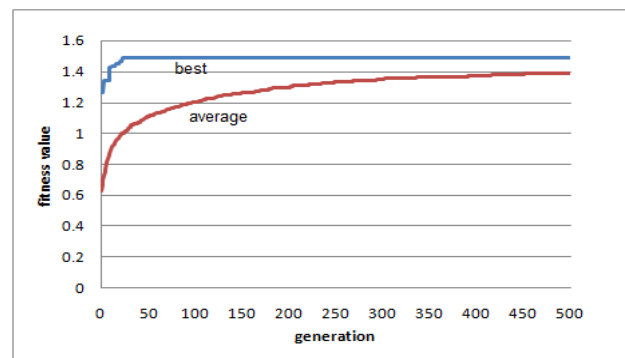

Fig. 1 The best and average fitness value

TABLE IV
SEQUENCE OF SELECTED PART TYPES

part type	value	chosen machines		
7	80	3	3	
5	160	1	2	
8	200	1	2	
3	60	3	3	1
Throughput	500			

TABLE V
MACHINES WORKLOAD

mac	workload	unbalance	number of slots	used slot	tools assigned
1	4200	200	20	10	1 2 3
2	3600	400	15	13	3 4 5
3	3700	300	20	13	1 8 9 10
System unbalance	900				

VI. RESULT AND DISCUSSION

To evaluate the performance of the RCGA, we generate 12 test-bed problems as shown in Table VI. Problems 1 to 4 are considered as small size problems, problems 5 to 8 are medium size problems and problems 5 to 8 are large size problems. Lengths of scheduling period for all machines are same within each problem. The other randomly generated parameters are shown in Table VII.

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

TABLE VI
TEST-BED PROBLEMS

problem	num. of part types	num. of machines	num. of tool types	scheduling period
1	8	4	20	4000
2	8	5	25	4000
3	10	4	20	4000
4	10	5	25	4000
5	16	4	20	7000
6	16	5	25	7000
7	18	4	20	7000
8	18	5	25	7000
9	24	4	20	10000
10	24	5	25	10000
11	26	4	20	10000
12	26	5	25	10000

TABLE VII
RANDOMLY GENERATED PARAMETERS

Parameters	Range
tool slot capacity of each machine	40-60
number of copies of each tool type	2-(nMac-1)
number of slots required for each tool type	3-7
number of operations of each part type	2-(nMac)
batch size of each part type	40-60
value of each part type (dollar)	5-10
number of possible machines for each operation	1-3
processing time of each operation	20-40
number of tool types required for each operation	2-5

nMac: number of machines

The RCGA is implemented in Java and experiment is carried out on personal computer equipped with Intel® Core™ i3-380 processor working at speed 2.53 GHz. The first step in our experiment is determining the most suitable selection method for the RCGA. Four common selection methods (roulette wheel, binary tournament, elitist, and replacement) are examined. The other parameters are set as follows:

- Crossover rate is 0.25.
- Mutation rate is 0.05
- Population size is 500, 1000 and 1500 for small size problems, medium size problems and large size problems respectively.
- The weighted parameters are $\alpha_1=3$ and $\alpha_2=1$.
- GA iterations will be stopped after 5,000 successive generations no longer produces better results.

By using these parameters and data from problem 7, we run the RCGA 10 times and obtain results of minimum (F_{min}), maximum (F_{max}), and average (F_{avg}) of fitness values as shown in Table VIII. The average computation time (*seconds*) and number of iterations to obtain the best solution (*itr best*) are also presented. Here, the replacement selection method produces a higher of average of fitness value than other methods. Therefore, we use this selection method in the next step of the experiment.

TABLE VIII
COMPARISON AMONG SELECTION METHODS

selection	F_{min}	F_{max}	F_{avg}	time	itr best
roulette wheel	2.1264	2.3104	2.1875	58.5	3593
binary tournament	2.2250	2.4175	2.3488	24.7	1980
elitist	2.1962	2.4101	2.3344	26.7	1284
replacement	2.3331	2.4178	2.3659	38.9	7181

By using the replacement selection, the RCGA can maintain the population diversity and explore the search space better. It is indicated by its significantly higher number of iterations to obtain the best solution. In contrast, the other selection methods achieve their convergence faster which may indicate that they are trapped in local optimum areas and cannot obtain a better solution. Fig. 2 depicts a one run from each selection method. It shows the improvement of the best fitness value along generations. While all other selection methods achieve their convergence in less than 2000 generations, the replacement selection gradually improve its chromosomes to obtain higher fitness value.
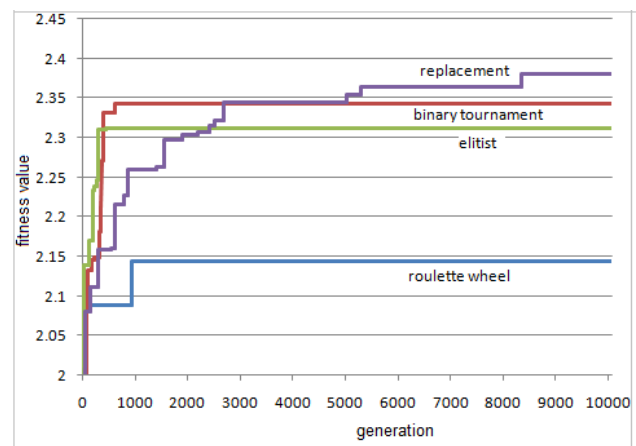

Fig. 2 The best fitness value for each selection method

The second step in our experiment is determining the most suitable crossover rate and mutation rate for the RCGA. Appropriate crossover rate and mutation rate will help the RCGA to balance its exploration and exploitation ability and avoid the premature convergence [26]. In order to get a fair result, we vary the crossover rate (*cr*) from 0 to 0.4 and set the mutation rate (*mr*) in such way that *cr+mr*=0.4. Here, all runs produce 0.4x1000 offspring in each generation. Again, we run the RCGA 10 times using problem 7. The result is presented in Table IX. Apparently, the best result is produced by using crossover rate of 0.3 and mutation rate of 0.1. Here, by using a low value of crossover rate the RCGA will mostly depend on its mutation rate and tend acting as a random search method. In other hand, the RCGA will lose its ability to maintain population diversity if using a high crossover rate and a low mutation rate. Inability to maintain population diversity means that the RCGA cannot explore the search space effectively and will likely be trapped in local optimum area.

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

TABLE IX
COMPARISON AMONG CROSSOVER RATES

crossover rate	mutation rate	average of fitness value
0.00	0.40	2.377
0.05	0.35	2.376
0.10	0.30	2.392
0.15	0.25	2.388
0.20	0.20	2.393
0.25	0.15	2.384
0.30	0.10	2.400
0.35	0.05	2.366
0.40	0.00	2.294

After determining the most suitable crossover rate and mutation rate for the RCGA, we run the RCGA for all test-bed problems. To measure the performance of the RCGA we use frequency of achieving optimum solution (FOS) and deviation of objective values resulted by GA to its optimum values. The optimum solutions are obtained by using branch-and-bound method. It is should be noted that branch-and-bound method required computational time more than 10 hours to solve particular test bed problems which cannot be accepted on daily operation of the FMS. Equation (16) shows the deviation of average fitness values from 20 runs of GA to optimum fitness value. F_{opt} is fitness value obtained by branch-and-bound method. FGA_r is fitness value obtained by GA in run r.

$$F_{dev} = \left| \frac{\left(F_{opt} - \left(\sum_{r=1}^{20} FGA_r\right)/20\right)}{F_{opt}} \right| \times 100\% \qquad (16)$$

The computational results are presented in Table X. Column 'time' shows average of computation time (seconds) from 20 runs of the RCGA. Columns 'F', 'TH' and 'SU' below column 'RCGA' show the average of fitness value, throughput and system unbalance obtained from 20 runs of the RCGA.

TABLE X
COMPUTATIONAL RESULTS

problem	RCGA					Optimum values			F_{dev}(%)
	FOS	Time (seconds)	F	TH	SU	F	TH	SU	
1	20	4.29	2.545	1,616.0	803.0	2.545	1616	803	0.00
2	20	5.09	2.926	2,591.0	9,838.0	2.926	2591	9838	0.00
3	20	5.24	2.972	3,058.0	6,858.0	2.972	3058	6858	0.00
4	20	6.26	2.531	2,196.0	3,233.0	2.531	2196	3233	0.00
5	12	16.47	2.133	2,604.0	**3405.6**	2.156	2,676	3,738	1.06
6	13	23.63	1.936	2,587.9	7,951.6	1.968	2,605	7,126	1.59
7	2	22.21	2.404	3,321.9	**3,543.1**	2.458	3,595	5,529	2.23
8	17	19.86	2.077	2,861.4	4,997.4	2.088	2,871	4,768	0.51
9	3	84.09	2.260	3,940.9	4,832.4	2.349	4,150	4,204	3.79
10	8	59.06	1.803	3,179.6	**10,666.6**	1.809	3,212	10,879	0.34
11	0	84.29	2.248	4,286.7	6,077.9	2.305	4,417	5,519	2.45
12	1	92.36	1.971	3,893.3	10,956.0	2.018	3,937	9,291	2.31
average		35.25							1.19

Apparently for a small number of part types (8 and 10), the proposed real coded GA could achieve optimum solution in all runs (problems 1 to 4). These results are obtained in less than 7 seconds. In the medium size problems (problems 5 to 8), the best result is obtained in problem 8 with F_{dev} of 0.51% and the worst solution is occurred in problem 6 with F_{dev} of 1.59%. Except for problem 7, the RCGA could produce optimum solutions in more than 10 runs for all problems.

The RCGA also obtains optimum solutions in several runs in the large size problems (problems 9, 10, and 12), the best result is obtained in problem 10 with F_{dev} of 0.34% and the worst solution is occurred in problem 9 with F_{dev} of 3.79%. Overall, in larger problems, F_{dev} values tend to increase as the search space becomes very wide. Increasing the population size, crossover rate and mutation rate will reduce F_{dev} values but the computation time will rise.

It should be noted that lower throughputs achieved by the RCGA is compensated by better (lower) system unbalances on problems 5, 7 and 10. All F_{dev} values are below 4% which may be regarded as good results considering these results are achieved in average of 35.25 seconds.

Note that these promising results are achieved by using only simple genetic operators. The novel proposed chromosome representation produces feasible solutions which minimize a computational time needed by GA to explore the feasible search space efficiently [35], [40]. Other approaches may require sophisticated strategies to achieve good results which may require excessive computation time such as hybridizing tabu search with simulated annealing [12], hybridizing genetic algorithm with simulated annealing [21], [29] and equipping particle swarm optimization with local search methods [14].

VII. CONCLUSION

The part type selection and loading problem with flexibilities of operations have been modeled in this paper. These NP-hard problems were solved by using real coded GA. Combination of proper representation and simple genetic operators could produce promising results in reasonable amount of time. By using 12 test bed problems, the proposed RCGA improves the FMS performance by considering two objectives, maximizing system throughput and maintaining the balance of the system (minimizing system unbalance). The

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

resulted objective values are compared to the optimum values produced by branch-and-bound method. The experiments show that the proposed RCGA could reach near optimum solutions in reasonable amount of time.

Further work will address more complex problem which considers alternative production plans which refer to possibility of producing part on alternative operation sequence. Resource allocation problem which refers to allocation of limited number of pallets and fixtures to the part types is also integrated to the existing problems. Therefore, a more powerful of GA is required. Hybridizing the RCGA with other heuristics methods and developing new crossover and mutation methods will be considered.

REFERENCES

[1] F.T.S. Chan and H.K. Chan, "A Comprehensive Survey and Future Trend of Simulation Study on Fms Scheduling," *Journal of Intelligent Manufacturing*, vol. 15, no. 1, pp. 87-102, 2004.

[2] I. Badr, "An Agent-Based Scheduling Framework for Flexible Manufacturing Systems," *International Journal of Computer, Information, and Systems Science, and Engineering*, vol. 2, no. 2, pp. 123-129, 2008.

[3] N. Kumar and K. Shanker, "A Genetic Algorithm for Fms Part Type Selection and Machine Loading," *International Journal of Production Research*, vol. 38, no. 16, pp. 3861-3887, 2000.

[4] Z. Binghai, X. Lifeng, and C. Yongshang, "A Heuristic Algorithm to Batching and Loading Problems in a Flexible Manufacturing System," *The International Journal of Advanced Manufacturing Technology*, vol. 23, no. 11, pp. 903-908, 2004.

[5] J. Venkateswaran, Y.-J. Son, and A. Jones. "Hierarchical Production Planning Using a Hybrid System Dynamic-Discrete Event Simulation Architecture," in *Proceedings of the 2004 Winter Simulation Conference*, 2004.

[6] M. Ben-Daya, "Fms Short Term Planning Problems: A Review," in *Manufacturing Research and Technology*, A. Raouf and M. Ben-Daya, Eds., Elsevier. p. 113-139, 1995.

[7] K.E. Stecke, "Design, Planning, Scheduling, and Control Problems of Flexible Manufacturing Systems," *Annals of Operations Research*, vol. 3, no. 1, pp. 1-12, 1985.

[8] S. Özpeynirci and M. Azizoglu, "Bounding Approaches for Operation Assignment and Capacity Allocation Problem in Flexible Manufacturing Systems," *Computers & Operations Research*, vol. 36, no. 9, pp. 2531-2540, 2009.

[9] S. Bilgin and M. Azizoglu, "Capacity and Tool Allocation Problem in Flexible Manufacturing Systems," *The Journal of the Operational Research Society*, vol. 57, no. 6, pp. 670-681, 2006.

[10] F.T.S. Chan and R. Swarnkar, "Ant Colony Optimization Approach to a Fuzzy Goal Programming Model for a Machine Tool Selection and Operation Allocation Problem in an Fms," *Robotics and Computer-Integrated Manufacturing*, vol. 22, no. 4, pp. 353-362, 2006.

[11] J.-H. Chen and S.-Y. Ho, "A Novel Approach to Production Planning of Flexible Manufacturing Systems Using an Efficient Multi-Objective Genetic Algorithm," *International Journal of Machine Tools and Manufacture*, vol. 45, no. 7–8, pp. 949-957, 2005.

[12] R. Swarnkar and M.K. Tiwari, "Modeling Machine Loading Problem of Fmss and Its Solution Methodology Using a Hybrid Tabu Search and Simulated Annealing-Based Heuristic Approach," *Robotics and Computer-Integrated Manufacturing*, vol. 20, no. 3, pp. 199-209, 2004.

[13] A.K. Choudhary, M.K. Tiwari, and J.A. Harding, "Part Selection and Operation-Machine Assignment in a Flexible Manufacturing System Environment: A Genetic Algorithm with Chromosome Differentiation-Based Methodology," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 220, no. 5, pp. 677-694, 2006.

[14] S. Biswas and S. Mahapatra, "Modified Particle Swarm Optimization for Solving Machine-Loading Problems in Flexible Manufacturing Systems," *The International Journal of Advanced Manufacturing Technology*, vol. 39, no. 9, pp. 931-942, 2008.

[15] S.G. Ponnambalam and L.S. Kiat, "Solving Machine Loading Problem in Flexible Manufacturing Systems Using Particle Swarm Optimization," *World Academy of Science, Engineering and Technology*, vol. 39, 2008.

[16] A. Prakash, N. Khilwani, M.K. Tiwari, and Y. Cohen, "Modified Immune Algorithm for Job Selection and Operation Allocation Problem in Flexible Manufacturing Systems," *Adv. Eng. Softw.*, vol. 39, no. 3, pp. 219-232, 2008.

[17] M.T. Tabucanon, D.N. Batanov, and S. Basu, "Using Simulation to Evaluate the Batching Approach to Part Type Selection in Flexible Manufacturing Systems," *Integrated Manufacturing Systems*, vol. 9, no. 1, pp. 5-14, 1998.

[18] H.-W. Kim, J.-M. Yu, J.-S. Kim, H.-H. Doh, D.-H. Lee, and S.-H. Nam, "Loading Algorithms for Flexible Manufacturing Systems with Partially Grouped Unrelated Machines and Additional Tooling Constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 58, no. 5, pp. 683-691, 2012.

[19] K. Seok Shin, J.O. Park, and Y. Keun Kim, "Multi-Objective Fms Process Planning with Various Flexibilities Using a Symbiotic Evolutionary Algorithm," *Computers and Operations Research*, vol. 38, no. 3, pp. 702-712, 2011.

[20] M.K. Tiwari, S. Kumar Jha, and R. Bardhan Anand, "Operation Allocation and Part Type Selection in E-Manufacturing: An Auction Based Heuristic Supported by Agent Technology," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 4, pp. 312-324, 2010.

[21] M.K. Tiwari, S. Kumar, S. Kumar, Prakash, and R. Shankar, "Solving Part-Type Selection and Operation Allocation Problems in an Fms: An Approach Using Constraints-Based Fast Simulated Annealing Algorithm," *IEEE Transaction on Systems, Man, and Cybernetics—Part A: Systems and Humans*, vol. 36, no. 6, pp. 1170-1184, 2006.

[22] W. Shen, "Genetic Algorithms in Agent-Based Manufacturing Scheduling Systems," *Integr. Comput.-Aided Eng.*, vol. 9, no. 3, pp. 207-217, 2002.

[23] M. Tiwari, J. Saha, and S. Mukhopadhyay, "Heuristic Solution Approaches for Combined-Job Sequencing and Machine Loading Problem in Flexible Manufacturing Systems," *The International Journal of Advanced Manufacturing Technology*, vol. 31, no. 7, pp. 716-730, 2007.

[24] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*, New York: John Wiley & Sons, Inc., 2000.

[25] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, Berlin Heidelberg: Springer, 2006.

[26] M. Lozano and F. Herrera, "Fuzzy Adaptive Genetic Algorithms: Design, Taxonomy," *Soft Computing*, vol. 7, pp. 545–562, 2003.

[27] W.F. Mahmudy, R.M. Marian, and L.H.S. Luong. "Solving Part Type Selection and Loading Problem in Flexible Manufacturing System Using Real Coded Genetic Algorithms – Part I: Modeling," in *International Conference on Control, Automation and Robotics*. Singapore: World Academy of Science, Engineering and Technology. 773-779, 2012.

[28] W.F. Mahmudy, R.M. Marian, and L.H.S. Luong. "Solving Part Type Selection and Loading Problem in Flexible Manufacturing System Using Real Coded Genetic Algorithms – Part Ii: Optimization," in *International Conference on Control, Automation and Robotics*. Singapore: World Academy of Science, Engineering and Technology. 778-782, 2012.

[29] M. Yogeswaran, S.G. Ponnambalam, and M.K. Tiwari, "An Efficient Hybrid Evolutionary Heuristic Using Genetic Algorithm and Simulated Annealing Algorithm to Solve Machine Loading Problem in Fms," *International Journal of Production Research*, vol. 47, no. 19, pp. 5421-5448, 2009.

[30] M. Denizell and S. Sayin, "Part-Types Selection in Flexible Manufacturings Systems: A Bicriteria Approach with Due Dates," *Journal of the Operational Research Society*, vol. 49, pp. 659-669, 1998.

[31] D. Pacciarelli. "Loading Parts and Tools in a Flexible Manufacturing System," in *Proceedings of the 6th IEEE Mediterranean Conference on Control and Systems*. Alghero, Italy, 1998.

[32] S.K. Mukhopadhyay, S. Midha, and V.M. Krishna, "A Heuristic Procedure for Loading Problems in Flexible Manufacturing Systems," *International Journal of Production Research*, vol. 30, no. 9, pp. 2213, 1992.

[33] M. Liang, "Integrating Machining Speed, Part Selection and Machine Loading Decisions in Flexible Manufacturing Systems," *Computers & Industrial Engineering*, vol. 26, no. 3, pp. 599-608, 1994.

[34] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, New York: John Wiley & Sons, Inc., 1997.

[35] R.M. Marian, L.H.S. Luong, and K. Abhary, "A Genetic Algorithm for the Optimisation of Assembly Sequences," *Comput. Ind. Eng.*, vol. 50,

World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

no. 4, pp. 503-527, 2006.

[36] F. Herrera, M. Lozano, and J.L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review*, vol. 12, pp. 265–319, 1998.

[37] R.M. Marian, L. Luong, and S.D. Dao, "Hybrid Genetic Algorithm Optimisation of Distribution Networks—a Comparative Study," in *Intelligent Control and Innovative Computing*, S.I. Ao, O. Castillo, and X. Huang, Eds., Springer US. p. 109-122, 2012.

[38] N.J. Radcliffe, "Equivalence Class Analysis of Genetic Algorithms," *Complex Systems*, vol. 5, no. 2, pp. 183–205, 1991.

[39] H. M¨uhlenbein and D. Schlierkamp-Voosen, "Predictive Models for the Breeder Genetic Algorithm; Continuous Parameter Optimization," *Evolutionary Computation* vol. 1, pp. 25–49, 1993.

[40] R.M. Marian, L.H.S. Luong, and R. Akararungruangkul, "Optimisation of Distribution Networks Using Genetic Algorithms. Part 2: The Genetic Algorithm and Genetic Operators," *International Journal of Manufacturing Technology and Management*, vol. 15, no. 1, pp. 84-101, 2008.