International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **3(A)**, March **2012**

 ISSN 1349-4198
pp. 1591–1609

CONSTRAINT-CHROMOSOME GENETIC ALGORITHM FOR FLEXIBLE MANUFACTURING SYSTEM MACHINE-LOADING PROBLEM

Umi Kalsom Yusof[1], Rahmat Budiarto[2] and Safaai Deris[3]

[1]School of Computer Sciences
Universiti Sains Malaysia
1800 USM, Penang, Malaysia
umiyusof@cs.usm.my

[2]InterNetWorks Research Group, School of Computing
College of Arts and Sciences
Universiti Utara Malaysia
Sintok, Malaysia
rahmat@uum.edu.my

[3]Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 Skudai, Johor, Malaysia
safaai@utm.my

Received December 2010; revised June 2011

Abstract. *Manufacturing industries are facing a rapidly changing market environment characterized by product competitiveness, short product life cycles, and increased product varieties. This scenario has given rise to the demand for improved capacity planning efficiency while maintaining their flexibilities. One important aspect of capacity planning is machine loading, which is known for its complexity encompassing various types of flexibility aspects that pertain to part selection and operation assignment along with constraint. The main objective of flexible manufacturing system (FMS) is to balance the productivity and flexibility of the production shop floor. From the literature, researchers have proposed many methods and approaches to attain a balance in exploring (global improvement) and exploiting (local improvement). We propose a constraint-chromosome genetic algorithm to solve this problem, which aims at mapping the right chromosome representation to the domain problem as well as helps avoid getting trapped in local minima. The objective functions are to minimize the system unbalance and increase throughput while satisfying the technological constraints. The performance of the proposed algorithm is tested on 10 sample problems available in the FMS literature and compared with existing solution methods Based on the results, the overall combined objective function increased by 3.60% from the previous best result.*

Keywords: Flexible manufacturing system, Machine loading, System unbalance, Throughput, Genetic algorithm

1. **Introduction.** Flexible manufacturing systems (FMSs) have been developed to integrate computer-controlled configurations of numerical control machine tools, along with other auxiliary production equipment, and a material handling system to simultaneously manufacture low to medium volumes of various high-quality products at a competitive cost [1]. Given that FMSs are very expensive, it is crucial to manage them effectively to achieve optimum results with least investment risk; this largely depends on how the decision is being made to tackle the problems in FMS. One of the main objectives of FMS

is to achieve efficiency of a well-balanced transfer line while retaining the flexibility of the job shop [2].

One of the important aspects of capacity planning in FMS is specific to the tasks of assigning the machines, operations for selected part types, and the necessary tools to perform these operations within the technological constraints in order to maximize throughput and minimize system unbalance [2, 3]. In studying the production-planning problem, [4] concluded that there are two crucial sub-problems, namely, part selection and machine loading. The inputs to the decision pertaining to loading problems are received from the preceding decision levels, such as grouping of resources, selection of part mixes, and aggregate planning. These outputs can generate inputs to the next decision related to scheduling resources, dynamic operations, and control. Therefore, it is clear that the loading decision acts as an important link between strategic and operational level decisions in manufacturing.

Although part sequence selection, machine loading and tool configuration are different, they are very closely interlinked. The shared problems, such as part-tool machine compatibility, machine time availability and tool magazine capability are actually interrelated constraints. Nevertheless, primarily because of the complexity of solving the problems simultaneously, many previous studies have treated these components separately or rather sequentially. Nevertheless, [5] recommended an integrated approach to resolve the part selection and machine-loading problem. They concluded that an integrated approach provides good results compared with de-contextualized solutions, in which the objectives are conflicting in nature, thus leading to infeasible or non-optimal solutions.

In the present work, we consider a machine-loading problem in an FMS environment. The machine-loading problem refers to how different components and tools can be assigned and allocated, respectively, so that some measure of productivity is optimized given a set of part types to be produced, a set of tools required for processing the parts on a set of machines, and a set of resources, including material handling systems, pallets and fixtures [6]. The main objectives are maximization of throughput and minimization of system unbalance; these are treated as functions to solve the machine-loading problem while simultaneously considering the technological constraints, such as available machining time and tool slots for each machine. The functions are able to minimize the idle time of the machine, thus leading to maximal machine utilization and improvement of the overall system output. We propose a constraint-chromosome Genetic Algorithm (CCGA) to create a robust and effective algorithm that is able to find a feasible solution, with a better and more accurate result. Additionally, this algorithm is also able to achieve fast convergence and helps avoid being trapped in local optima.

The remainder of the paper is organized into the following sections. Section 2 discusses related works on the issue being discussed. Section 3 describes the problem and model formulation. Section 4 discusses the solution proposed to solve the problem. Section 5 discusses the experimental results of the proposed solution and compares it with those in previous literature. The last section concludes the paper.

2. **Related Work.**

There are many objectives related to machine-loading problems. Stecke [2] outlined machine-loading problems in detail and concluded its six main objectives. Due to the number of objectives covered involving the simultaneous determination of various factors (e.g., sequence, throughput and workload balancing) and the requirement to satisfy various technological constraints, machine-loading problem lies under the broad category of NP-hard problem [7]. The literature concludes that the problems pertaining to machine loading cover many objectives [2, 8, 9, 10, 11].

Basically, the machine-loading problem can be solved using two approaches: (a) optimization-based methods and (b) heuristic-oriented method. Studies on optimization-based methods include those of Stecke [2], Shanker and Tzen [9], Sarin and Chen [12], Nayak and Acharya [13], whose approaches involved integer programming, dynamic programming, branch, and bound. The researchers who proposed the heuristic-oriented approach are Mukhopadhyay et al. [10] who developed the concept of essentially ratio for minimizing the system unbalance that, in turn, can maximize the throughput. Mukhopadhyay et al. [14] developed the "modified insertion scheme" using simulated annealing, whereas Shanker and Srinivasulu [15] used heuristic approach to develop a two-stage branch and backtrack procedure. Tiwari et al. [11] proposed a heuristic approach and the Petri net model to delineate its graphical representation and validation. Sarma et al. [16] proposed the Tabu search-based algorithm with constraints of tool slots and machine time, while Swarnkar and Tiwari [17] worked on a hybrid Tabu search and simulated annealing. Finally, Nagarjuna et al. [18] suggested a heuristic-based multistage programming.

Most of the mathematical programming-based approaches and heuristic techniques are unable to produce good-quality optimal or near-optimal solutions to machine-loading problems because of their limitations. Although optimization-based methods are robust in their applicability, they tend to become impractical when the problem size increases. On the other hand, heuristic approaches are mainly dependent on rules and constraints of individual problems. For this reason, the heuristic approach has always faced difficulty in estimating results in a changed problem or environment. These limitations have motivated researchers to enhance the method and look for innovative searching techniques. One of the popular approaches is genetic algorithm (GA), which is famous for its ability to perform intelligent probabilistic searching. Li et al. [19] proposed GA to handle problems with multi-period and multilevel capability balancing issues, whereas Tiwari and Vidyarthi [20] used GA to allocate resources in ordering and increasing equipment utilization and throughput. Ip et al. [21] used GA in proposing a model to solve the planning and scheduling problem within a multiproduct production environment. Pongcharoen et al. [22] suggested the use of GA-based scheduling tool to handle multiple resource constraints and multiple level product structure. Kumar et al. [6] proposed constraint-chromosome GA in handling complex constraints in the FMS-loading problem.

GA is also being studied and applied to many other areas. For instance, Lin and Tsai [23] used two-stage GA to solve transportation problem. Maeda and Li [24] studied ways to optimize the GA search capability tuned with fuzzy adaptive search method, whereas Lin [25] proposed GA to solve fuzzy equations with constraints. Okuhara et al. [26] used GA to control worker and workload assignment in project management. Kim and Kim [27] suggested the use of GA to minimize the inter-cell interference in wireless communication systems. Suryoatmojo et al. [28] proposed GA to determine the optimal capacities of a PV system, battery bank, and diesel generator. Tan et al. [29] suggested a GA with redundancy-saving strategy for synchronous direct-sequence code division multiple access system.

Integrating GA into other algorithms or approaches has also been proposed in previous studies. Che [30] used hybrid GA to enhance productivity while considering product configuration change. Ono et al. [31] proposed a hybrid multi-objective GA and quasi-Newton method to find robust optimal solutions to enhance the local search facility and reduce search cost. Chen et al. [32] proposed a guided mimetic algorithm to solve scheduling problems.

The literature clearly shows that GA has proven to be robust in handling many types of manufacturing optimization problems. For the present research, we propose to construct the chromosome using a sequence of genes that have been created based on the constraint

of the machines and operations. Our objective functions focus on the maximization of throughput and the minimization of system unbalance. These objective functions will lead to the minimization of system idle time that, in turn, promotes higher machine utilization. In addition, machine-loading policy aims to maximize total system output, which is interpreted as throughput.

3. FMS Machine-Loading Problem.

3.1. Problem description.

The machine-loading problem in FMS can be represented as part type allocation with multiple machines and fixed number of slots. Given a planning horizon, part types may be scheduled randomly. The respective operation times and tool slot requirements for each machine are well known [9, 10, 11, 20]. A part type selection and machine-loading arrangement constitute two major components of strategic planning problem of FMS. Part type selection deals with the selection of a set of part types to be manufactured during the upcoming planning horizon, whereas the loading problem is concerned about the allocation of operations and the required machine and tools for the selected part types. Most of the earlier studies have proposed these problems separately because of their complexity; hence, the solution of part type selection problem may lead to an infeasible result for loading problem. In practice, part type selection is carried out at the beginning of the planning period, whereas allocation of operation and machine tools is done at a later stage. A part type requires one or more operations, with each operation performed by one or more corresponding machines. These parameters are set by the production requirement, which includes part type, number of operations for each part type and respective machining time, and number of tool slots required for each operation of each part type. The details of such information are planned and set in advance.

There are two types of operations complementary to each part type. Essential operations of a part type mean that these operations can be performed only on a particular machine using a certain number of tool slots, whereas optional operations imply that they can be carried out on a number of machines with the same or varying processing time and tool slots. Given that essential operation requirements can only be performed at specific machine(s), the flexibility lies in the selection of machines that are set for optional operations, in which there are more rooms for improving the machine allocation to yield better results. During planning horizon, machine-loading problem deals with a given part type and respective operations to appropriate machines, during which technological and capacity constraints must be considered. Therefore, it is logical to ensure that optional operations consider all possible routes. To understand the complexities of the machine-loading problem for a random FMS, we take an example of eight part types, which are scheduled for processing on four machines. Each machine has five tool slots with different processing times for different operations as well as different processing times when run on different part types. Each part type consists of four operations that are performed on any machine while maintaining the sequence of the operations. The flexibility of each machine to perform many different operations while allowing several operation assignment possibilities can then generate alternative part routes.

To summarize the machine-loading problem requirements, Shanker and Tzen [9] and Shanker and Srinivasulu [15] summed up the above features of FMS machine-loading problem and developed nine more similar problems describing several features related to batch size of part type, number of operations, machine, tool slot requirement and unit processing time for proper machine allocation. The assumptions made before proceeding with the modeling of the machine-loading problem are as follows [11]:

- Non-splitting of part type-this implies that a part type undertaken for processing is to be completed for all its operations before considering a new part type;
- Unique part type routing-although flexibility exists in the selection of a machine for optional operation, the operation must be completed on the same machine once a machine is selected;
- Sharing of tool slots is not considered; and
- Parts are readily available.

3.2. **Model formulation.** The objective functions for the present research are the maximization of throughput and minimization of system unbalance. These are to be combined in order to attain the overall function for machine-loading problem. It is crucial to combine both objective functions in a logical manner that also considers the technical constraints. The problem described in the previous section is mathematically described in this section. The notations used to demonstrate the objective functions are shown below.

Subscripts

i	part type, $i = 1, \ldots, N$
m	machine, $m = 1, \ldots, M$
t	tool type, $t = 1, \ldots, T$
j	operation type, $j = 1, \ldots, J$

Parameters

H	total planning horizon (480 minutes in the example)
UT_m	underutilized time on machine m
OT_m	overutilized time on machine m
β_i	batch size of part type i
α_i	$= 1$ if part type i is selected, otherwise 0
t^a_{im}	time available on machine type m after allocation of operation j of part type i
t_{imj}	time required by machine type m for operation j of part type i
T^r_{im}	number of machines for each machine type remaining on machine m after allocation of operation j of part type i
T^a_{im}	number of machines for each machine type available on machine m after allocation of operation j of part type i
T_{imj}	number of machines for each machine type required by machine m for operation j of part type i
Y_ij	set of machine type on which operation j of part type i can be performed

3.3. **Formulation of objective function and constraints.** The objective functions for the present research can be formulated as listed below.

(i) The first objective is to minimize the system unbalance in order to maximize system utilization. Thus, the first objective function can be formulated as follows:

$$Minimize \sum_{m=1}^{M} (UT_m - OT_m), \tag{1}$$

which is equivalent to

$$Maximize - \sum_{m=1}^{M} (UT_m - OT_m). \tag{2}$$

The above is also equivalent to:

$$F1 = \frac{M * H - \sum_{m=1}^{M}(UT_m - OT_m)}{M * H}. \tag{3}$$

(ii) The second objective function is to maximize throughput in order to maximize system efficiency. Thus, the second objective function can be stated as follows:

$$F2 = \frac{\sum_{i=1}^{N} \beta_i * \alpha_i}{\sum_{i=1}^{N} \beta_i}. \tag{4}$$

(iii) The third objective function is a combination of both of the above objective functions and is given by:

$$F = \frac{M * H - \sum_{m=1}^{M}(UT_m - OT_m)}{M * H} + \frac{\sum_{i=1}^{N} \beta_i * \alpha_i}{\sum_{i=1}^{N} \beta_i}. \tag{5}$$

These two objective functions may be assigned to the weights accordingly, after which the researcher decides which objective should carry more weight. In the present study, to simplify the computation, we equally assigned the weight for each objective function, such that both carry a value of 1.

The above objective functions are subjected to the following constraints:

(i) System unbalance: This is equivalent to the sum of the idle time remaining on machines after allocation of all feasible part types. The value of machine utilization must be either zero (100% utilization of system) or a positive value. The constraint may be expressed as follows:

$$\sum_{m=1}^{M} UT_m - OT_m \geq 0. \tag{6}$$

(ii) Non-splitting of part type: This constraint is to ensure that once a part type is considered for processing, all the operations under that part type must be completed first before undertaking a new part type. The constraint may be expressed as follows:

$$\sum_{m=1}^{M} \sum_{j=1}^{J} \alpha_{imj} = \alpha_i * J, \ i = 1, 2, \ldots, N. \tag{7}$$

(iii) The available time on machines: This should be greater than or equal to the time required by the next part type to be assigned to this machine and is expressed as follows:

$$\sum_{j=1}^{P} t_{imj}\alpha_{imj} \leq t_{im}^{a}, \ i = 1, 2, \ldots, N. \tag{8}$$

(iv) Unique part type routing: Although flexibility exists in the selection of a machine for an optional operation, the operation must be completed on the same machine and expressed once a machine is selected; this is expressed as follows:

$$\sum_{m \in Y_{ij}} \alpha_{imj} \leq 1, \ i = 1, 2, \ldots, N; \ j = 1, 2, \ldots, J. \tag{9}$$

(v) The number of machines for each machine type and the remaining time on any machine after any assignment of part type should always be positive or zero.

$$T_{imj}\alpha_{imj} \geq 0, \ i = 1, 2, \ldots, \ m = 1, 2, \ldots, M, \tag{10}$$

$$t_{imj}\alpha_{imj} \geq 0, \ i = 1, 2, \ldots, \ m = 1, 2, \ldots, M. \tag{11}$$

(vi) The integrity of decision variables: The decision variables possessing a value of 0 and 1 integers are as follows:

$$\alpha_i = \begin{cases} 1 & \text{if part type } i \text{ is selected} \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

$$i = 1, 2, \ldots, N$$

$$\alpha_{im} = \begin{cases} 1 & \text{if part type } i \text{ is assigned to machine } m \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

$$i = 1, 2, \ldots, N;\ m = 1, 2, \ldots, M$$

$$\alpha_{imj} = \begin{cases} 1 & \text{if operation } j \text{ of part type } i \text{ is assigned to machine } m \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

$$i = 1, 2, \ldots, N;\ m = 1, 2, \ldots, M;\ j = 1, 2, \ldots, J$$

4. **The Proposed CCGA.** GA is a population-based method, which is inspired by the principle of natural evolution [33]. GA is motivated by the mechanism of natural selection, the biological process in which selection is done based on the fittest individuals represented as strings of bits that are analogous to chromosomes and genes. Using the iteration process, GA produces good genes by eliminating the bad ones. The crossover operator aims to combine the best traits of parents to produce better offspring, whereas in another operator, mutation is used to introduce random modifications of the chromosome to add diversity of the population as well as to prevent GA from becoming trapped in finding good but non-optimal solutions. GA works with the idea that stronger individuals are likely to be winners who can survive in a competitive environment. It uses the direct analogy of the natural evolution, in which, starting from a population of chromosomes, fitter chromosomes tend to yield good-quality offspring, indicating that they create better solutions.

GA is capable of handling many types of manufacturing optimization problems, whereas other techniques are able to solve certain types of problems only. For example, Saravanan [34] presented many extensive examples and comparisons of AI-based techniques (i.e., GAs, Tabu search, simulated annealing, particle swarm optimization, and ant colony optimization) that are applied to handle several manufacturing optimization problems. In addition, it also offers a good potential of developing a strategy to handle problem constraints within the GA framework that enhances the solution of machine-loading problems for FMS. Although GA is known as a global search technique (exploitative), it has the tendency to converge toward local optima and often fails in finding the global optimum of the problem [35]. In other words, it has a tendency to focus on the local optimum rather than explore new solutions to achieve global optimum. Due to this tendency, a good strategy has to be adopted to transform GA into an exploitative as well as an explorative algorithm.

The effectiveness of GA highly depends on how well it can handle problem constraints. There are numerous techniques that have been developed to handle constraints. Michalewicz and Nazhiyath [36] surveyed several constraint methods for GA, whereby the most common approach is to add a penalty function to the objective function to transform a non-constraint problem into a constraint one. In this penalty function, the basic rule is to add no penalty when all the constraints have been satisfied, and non-zero penalty otherwise. Nevertheless, quantifying constraint violation is not always practical, especially when we are dealing with nonnumeric variables. Therefore, applying penalty functions directly onto the FMS framework can be a tedious task. Cormier et al. [37] proposed constraint-based GA to solve problems in concurrent engineering. This concept is adopted in our current work to solve FMS problem, with some modifications. In our work, we

consider constraints during the formation of chromosome, which we call CCGA. The model formulation we used is based on Yusof and Deris [38]; however, owing to the different problem domains, the gene construction within the chromosome differed. In Yusof and Deris [38], the gene is set as a sequence of machine for individual part type, where more than one machine type may be allocated to each part type. The allocation of the machines is denoted as bit value 1 and 0 otherwise. On the other hand, for the present work domain problem, each gene was based on the operation and machine number that was assigned to the part type.

For our proposed work, the machine pool is created first to store all possible genes to be seeded for the formation of the chromosome. The machine pool for each part type is generated based on the part type schedule, along with the operations to be performed with the potential machines for that particular operation. Each part type is assigned the gene(s) that consisted of part type with the respective operation and possible machines. For example, part type 1 only has one operation with only one possible machine (essential operation); thus, the gene is labeled 113; in comparison, for part type 6, operation 2 has possible genes of 624, 622 and 623 because this operation is allowed to be processed on three machines (machines 4, 2 and 3). Machine pool provides the gene selection, which creates feasible allocation for each part type. The chromosome representation is constructed based on the genes generated in this machine pool. This approach ensures that a feasible solution can achieve fast convergence, with more accurate and better results. The example of machine pool for problem set 1 is shown in Table 1.

Table 1. Machine pool for problem set 1

Part Type	Operation Number	Possible Machines	Possible Gene		
1	1	3	113		
2	1	1,4	211	214	
	2	4	224		
	3	2	232		
3	1	4,1	314	311	
	2	3	323		
4	1	3	413		
	2	4	424		
5	1	2,3	512	513	
	2	2	522		
6	1	4	614		
	2	4,2,3	624	622	623
	3	2,1	632	631	
7	1	3,2,4	713	712	714
	2	2,3,1	722	723	721
	3	4	734		
8	1	1,2,3	811	812	813
	2	2,1	822	821	
	3	1	831		

Initial GA populations are generated using a random function, in which the sequence of part type, called part sequence, is randomly generated from the part type scheduled for the processing. The sequence determined the arrangement of the part type processing, and as part of the constraint, the same part type cannot exist twice in the chromosome string (i.e., no duplication of part types). On the basis of the part sequence, the chromosome, called part-operation chromosome, is generated. The genes are randomly selected from

the machine pool for each part type. Although the genes are randomly selected, the characteristic values of the constraints are applied to the genes in the machine pool, in which each chromosome created produced a feasible solution.

To enhance the possibility of creating more solutions, we create new populations based on the selection criteria. Genetic operations, crossover and mutation, are applied to these populations, after which the fitness function is calculated. The constraint is also imposed for crossover and mutation operations in generating new population in order to make the search space wider while maintaining the solution feasibility. Afterwards, these populations are assigned a fitness value based on the fitness function. These steps (i.e., the formation of new solution until fitness calculation) are repeated until the satisfactory result is achieved. The flowchart of the proposed model is described in Figure 1.

4.1. **Chromosome representation and initialization.** Chromosome representation is a way of representing how a domain problem being mapped and significantly affects the GA performance. Each individual represents a solution to the problem, and an improper or wrong representation of the chromosome structure often leads to poor performance [39]. It may also lead to the inability to find the right solution. Basic GA has a tendency to produce a number of infeasible chromosomes in each generation because of randomized crossover and mutation [33]. Thus, the chromosomes must be "repaired" to ensure feasible solution. This process lengthens the computational time as well as produces poor performance. Thus, it is crucial to come up with a good chromosome representation design. The main objective is to generate chromosomes that can produce feasible solutions after each crossover and mutation. Ho and Tay [40] focused on good feasible chromosomes that can reduce the search space as well as make the algorithm more efficient, hence reducing computational time.

An individual population can consist of one or more chromosomes and may be represented by strings. The selection of a string format for the individual is the first and a very important step to achieving successful implementation of GA. For this reason, we strive to find an efficient coding of each individual while considering all the constraints of the machine-loading problem. In our coding, each individual consists of part-sequence and part-operation chromosome. The part sequence defines the sequence of the part type to be allocated, and the part-operation chromosome is the respective sequence of part type with each operation to be performed on a selected machine. The genes of the respective chromosome describe a concrete allocation of machines to each operation as well as the sequence of machines to be run on each operation.

A sample of part sequence for Table 2 (problem set 1) is shown in Figure 2. The part types are arranged based on random selection from the scheduled part types, where each part type is checked against its existence to ensure that no same part type is generated in the sequence. The part-operation chromosome is then created based on this sequence. To ensure the feasibility of the solution, each part type in the sequence retrieves the respective operations that are assigned to it. Given that the operation sequence for each part type has to follow the sequence (i.e., the first operation needs to be allocated first before the second one), there is a requirement to maintain such a sequence.

The convergence speed of a GA is crucial in finding a global optimum or an acceptable solution. The reproduction strategy is in a position to influence the convergence speed of the algorithm. With these criteria in mind, we assigned each unit a combination number of part type, operation, and machine. Afterwards, we set the chromosome to a sequence of these combinations. Part-operation chromosome is created based on the number of operations required for each part type. The feasible machines for each operation are randomly retrieved from machine pool based on the machine pool index; this index kept

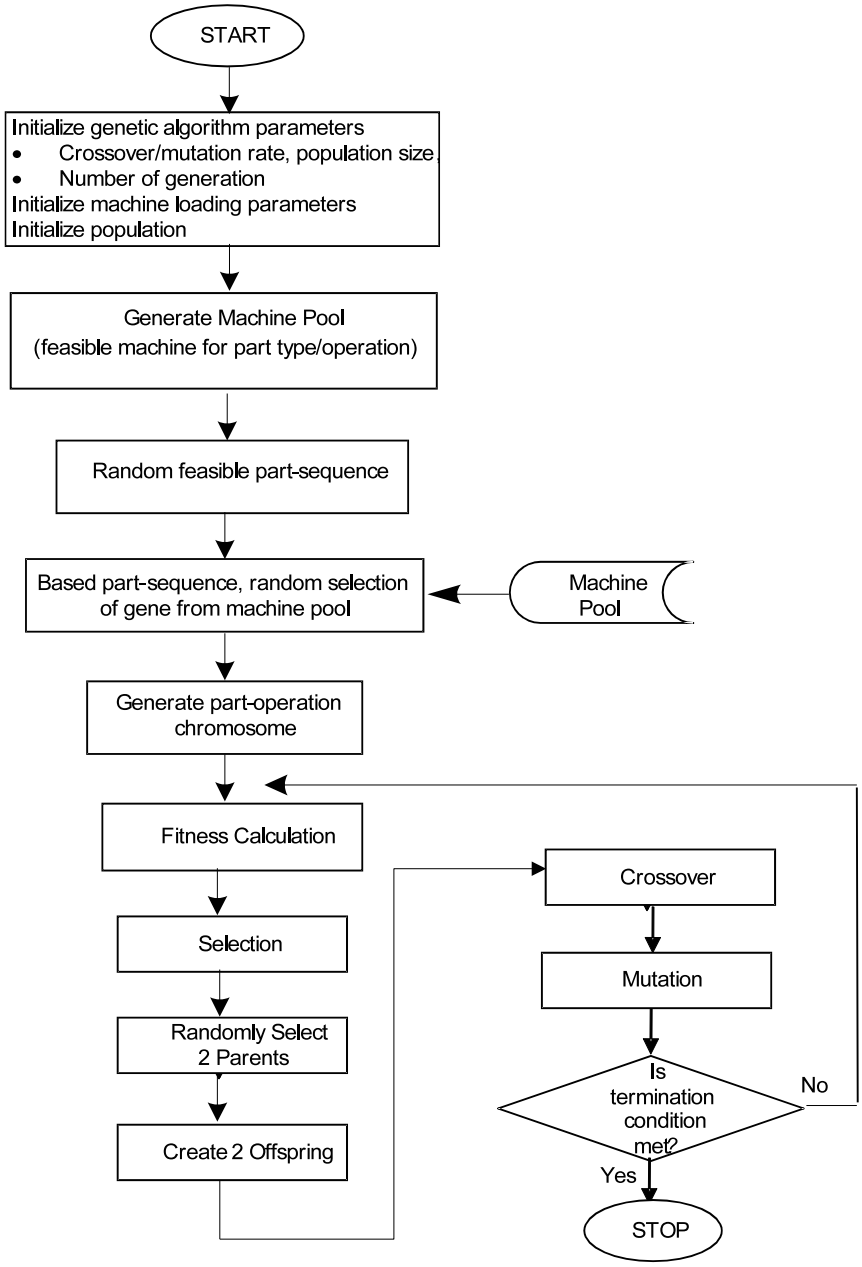

FIGURE 1. Flow chart of the proposed model

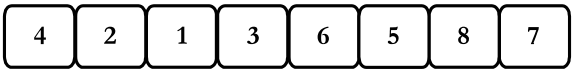

FIGURE 2. A sample of part sequence

track of the values for the respective part and operation. For example, for a part type with two operations, two genes are created for that particular part type, three genes for three operations, and so on. Figure 3 shows two examples of a possible value of part-operation chromosome based on the part sequence (4 2 1 3 6 5 8 7). Since the operation of part type 4 is considered as an essential operation (both on operations 1 and 2), whereby the operation can be performed on a particular machine only, there is no change in the gene

TABLE 2. Description of the problem (adapted from [10])

Part	Batch Size	Number of Operation	Operation Number	Machine Number	Unit Proc Time (minutes)	Tool Slot	Total Proc Time
1	8	1	1	3	18	1	144
2	9	3	1	1,4	25	1	225
			2	4	24	1	216
			3	2	22	1	198
3	13	2	1	4,1	26	2	338
			2	3	11	3	143
4	6	2	1	3	14	1	84
			2	4	19	1	114
5	9	2	1	2,3	22	2	198
			2	2	25	1	225
6	10	3	1	4	16	1	160
			2	4,2,3	7	1	70
			3	2,1	21	1	210
7	12	3	1	3,2,4	19	1	228
			2	2,3,1	13	1	156
			3	4	23	3	276
8	13	3	1	1,2,3	25	1	325
			2	2,1	7	1	91
			3	1	24	3	312

FIGURE 3. (a) Possible value of part-operation chromosome, (b) Another possible value of part-operation chromosome

that generated the part-operation chromosomes, as shown in Figures 3(a) and 3(b). On the other hand, operation 1 of part type 2 is optional operation performed on a number of machines with the same or varying process times and tool slots. In this example, operation 1 may be performed on machine 1 or 4; thus, total processing time for both may vary, leading to different results.

4.2. **Fitness evaluation.** After chromosome initialization, each sequence of the population is evaluated according to the objective functions set for the problem. The evaluation of the chromosome is a crucial part of GA, because the chromosomes selected for mating are based on their fitness. The fitness function then decides whether or not the machine allocation is good for the machine-loading problem. Machine-loading problem in FMS involves multiple objective functions, and as such, the model should be robust for the user to specify several objectives.

For the fitness calculation, the part-operation chromosome is generally used. This chromosome carries all relevant information to calculate the fitness value, including the operations required for each part type and the machine in which the operation is to be performed. From this information, the total processing time and machine slots for each chromosome can be determined and the fitness value can also be calculated. Using this information, we can also determine which part type in the string can be assigned. Afterwards, the fitness function determines the throughput and system unbalance. Throughput

is defined as units of part types produced for all selected part types during the planning horizon; here, the units are the sum of the batch sizes. System unbalance refers to the sum of the over- or underutilized time on all the machines after the allocation of all part types. The aim of the fitness function is to find the highest value of throughput and the lowest value of system unbalance. Both fitness functions are combined to create the combined objective function (COF). The higher the value of COF, the fitter the chromosome would be. In addition, because the chromosomes are constructed from valid genes from the machine pool, the result is expected to be valid.

4.3. **Selection.** The selection process is the driving force allowing the GA to determine the continuation of the evolutionary process flow. Previous studies have proposed, tested, and compared many selection methods. The present research proposes the use of a simple roulette wheel selection method, in which the parents for the next mating are chosen based on the fitness proportionate selection principle. In the proportionate selection, the fitness function assigns a fitness value to each chromosome in order to associate a probability of selection with each individual chromosome; thus, a higher fitness leads to improved chances of being selected. If f_i is the fitness of individual i in the population, its probability P of being selected is represented as follows:

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{15}$$

where N is the number of individuals in the population.

4.4. **Crossover.** Constructing genetic operators requires careful consideration in order to ensure that the crossover and mutation work proceed as expected. The chromosome has to be broken up to give the result, which is appropriate for our problem. Three crossovers are usually applied: partially mapped, order, and cycle crossover. Depending on how the chromosome represents the solution, sometimes a direct swap may not be possible or can create infeasible result. One such case is when the chromosome is an ordered list (e.g., the ordered list for the cities to be travelled in a travelling salesman problem), in which the same city should not be visited more than once.

Our chromosome, called part-operation chromosome, also falls into this category. Within such a category, the same gene cannot appear more than once, and direct swap between parents can lead to the duplication or deletion of certain genes, thus creating infeasible result. For this reason, we apply ordered chromosomes with two-point crossover. Two crossover points of the first parent are randomly generated to determine the first portion of the first offspring. The remaining genes of chromosome in the first offspring are then taken from the second parent, under the consideration that the same gene is not to be taken again.

Furthermore, the crossover application should consider genes that belong to the same part. The crossover point should consider taking the right portion, in which the whole part type can be moved together. Failure to do so can cause infeasible result and violate the condition that the part type operation should be finished first before allocating the next one. The two selected points consider the same number of part type for both parents. Due to the different number of operations per part types, the length of these two points differs from parents 1 and 2. The example in Figure 4 shows the crossover application. The first portion of offspring is taken from the randomly selected two points of the first parent, as denoted by the vertical line (part types 2, 1 and 3). The information after the crossover point is ordered because it is in the other parent. The second offspring (part types 4, 2 and 1) repeats the same process while creating the second offspring with a second parent. Then, the newly selected part types are randomly reassigned with the

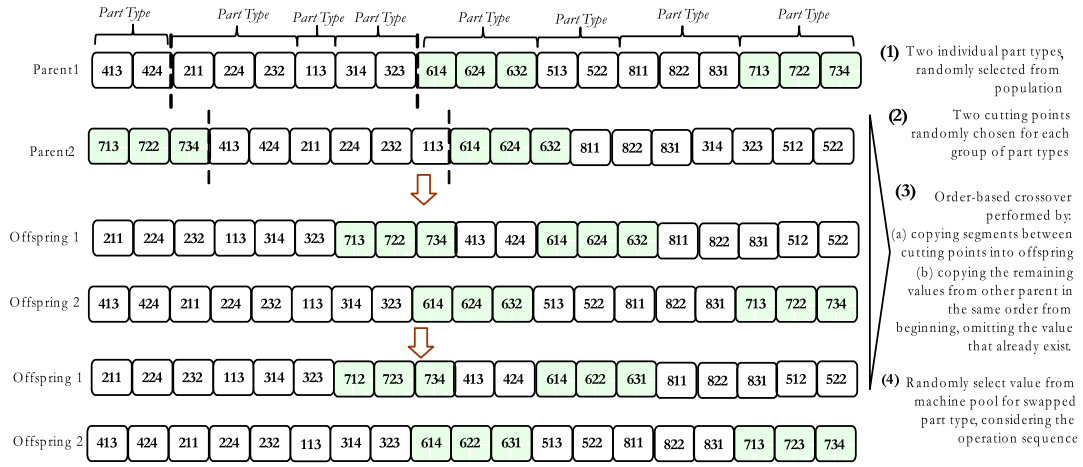

FIGURE 4. Ordered chromosome crossover

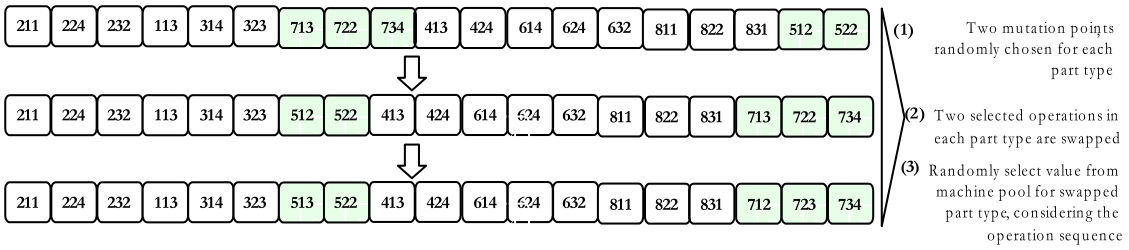

FIGURE 5. Mutation by reciprocal exchange

value from machine pool while maintaining the operation sequence for each part type so as to ensure diversity. Genes 713 and 722 can change to 712 and 723 (offspring 1), respectively, because the operations are optional operations; in comparison, gene 614 (offspring 2) remains the same because the operation is an essential operation.

4.5. **Mutation.** The purpose of mutation in GA is to ensure that the algorithm avoids the potential of being trapped in local minima by preventing the population of chromosomes from becoming too similar to each other. This phenomenon can eventually slow down the evolution process and prevent the ability of the algorithm to achieve good results. The same reason also explains why most of the GA algorithms avoid taking only the fittest of the population during selection for the next generation, but rather making the selection randomly (or semi-randomly) with weighing factors toward the fitter ones.

For mutation, we used reciprocal exchange mutation, in which two randomly selected part types are swapped in the same chromosome. This operation required ensuring that the genes of whole part type were swapped instead of the individual genes to avoid an unfeasible result. In the example shown in Figure 5, the mutation operator randomly selects two part types, the fourth and the eighth part types (genes 7-9 and 18-19, respectively), and those genes for the two part types are swapped. Then, to create diversity, we randomly reassign the newly selected part types with the value from machine pool while maintaining the operation sequence for each part type. This procedure is similar to the crossover operation.

TABLE 3. Machine properties

Machine Number	Processing Time (Minutes)	Number of Slot
1	480	5
2	480	5
3	480	5
4	480	5

4.6. **Termination condition.** The present research uses the number of generations as the termination condition. Time limit is not used because it is not the yardstick in this problem due to a relatively small data set. In addition, the fitness limit is unknown for the problem set; thus, it cannot be used as a termination condition as well. The number of generation set for the present research was 75, which is based on the size of the data sets as well as the generation number used by the previous research on the same data sets.

5. **Results and Discussion.** The proposed approach is implemented using C# compiler. It is tested on the benchmark problems available [2], which contains 10 data sets. Each data set has a different number of part types, number of operations, and machines that can be assigned to each operation. To illustrate the proposed algorithm, the problem set number 1 generated by Mukhopadhyay et al. [2] was adopted. The data are given in Table 2. There are eight part types to be loaded on four machines. All the machines are assumed to have one shift (i.e., 480 minutes of available time), and all of them have five tool slots each. The detailed information of the machines is shown in Table 3.

The number of generations required to converge largely depends on the effectiveness of the chromosome representation and the fitness measure, which evaluates the individual solution and the suitability of the operators. The time taken to find solutions is dependent on the size of chromosomes, the tightness of the constraints, the approach used in designing the chromosome representation, and the effectiveness of fitness function. This is because substantial processing time is required to filter non-feasible solutions. The work is expected to manage machine utilization and allow more part types that are capable of scheduling in current resources. Applying a constraint-based GA with the focus of effective chromosome representation has also been studied by Chen-Fang and Kuo-Ming [39], who found that an improper or wrong representation of chromosome structure may lead to poor performance as well as the inability to find the right solution.

In our present work, the chromosome representation ensures that the genes represent the right combinations that only allowed the valid machines to be allocated to the respective part types. Nevertheless, the fitness function still checked the chromosome to ensure that only the right combination and the total amount of time allocated did not exceed the remaining time available for the machine. The crossover and mutation operators performed their tasks effectively because of the effectiveness and efficiency of chromosome representation as well as the fitness function. The crossover operator then attempted to find all the permutations of machine allocation, after which the mutation operator provided the variety of parents that may include the good traits as the source of required traits for the next generation. In addition, the proper measurement is taken to ensure that the group of genes that belonged to the same part types are taken instead of the individual genes in performing the genetic operation.

The various combinations of parameter values are also used to conduct a sensitivity analysis on the sample data set to identify the optimal control parameters for the proposed CCGA, as shown in Table 4. Exhaustive computations are carried out to assess the

TABLE 4. Control parameters of the proposed algorithm

Control Parameter	Option		
	1	2	3
Population size	20	20	20
Number of generations	75	75	75
Selection operator	Roulette wheel	Roulette wheel	Roulette wheel
Crossover operator	Ordered chromosome	Ordered chromosome	Ordered chromosome
Mutation operator	Reciprocal exchange	Reciprocal exchange	Reciprocal exchange
Probability of crossover	0.7	0.7	0.7
Probability of mutation	0.01	0.1	0.3
Overall fitness	1.7871	1.7882	1.8045

TABLE 5. Summary of results of the proposed CCGA

Data Set	Part Types Assigned	Part Types Unassigned	Value of the Chromosome	Number of Generations to Converge
1	(6, 5, 7, 3, 1)	(4, 8, 2)	(614 622 632 512 522 712 721 734 311 323 113 413 424 812 822 831 214 224 232)	3
2	(2, 1,5,3,6)	(4)	(211 224 114 122 514 521 311 612 412)	3
3	(1, 3, 5, 2)	(4)	(112 314 323 514 523 212 411)	1
4	(5, 2, 4, 3, 1)	–	(511 214 223 412 423 432 314 112 121 134)	0
5	(2, 5,1, 6, 4)	(3)	(211 513 112 122 611 414 422 433 313 324)	1
6	(3, 2, 1, 5, 6)	(4)	(311 211 223 112 513 521 612 412)	3
7	(1, 6, 3, 5, 2)	(4)	(111 612 622 311 513 214 222 233 413 424)	1
8	(2, 5, 1, 7, 6)	(4, 3)	(213 514 523 112 123 714 722 413 424 613 621 631 314 324 332)	2
9	(6, 5, 7, 2, 4, 3, 1)	–	(614 621 514 521 712 724 213 413 424 312 113 121)	0
10	(3, 6, 2, 1, 4, 5)	(2)	(311 322 613 621 213 224 233 114 124 132 413 424 431 511 522 534)	2

effectiveness of the proposed algorithm. On the basis of the experiments, option 3 gives the overall best result: here, the higher probability of mutation rate improves the result. Higher mutation rate generally promotes higher probability of selecting a new gene and helps avoid being trapped in local optima. Although a higher mutation rate may lead to an unfeasible result in certain applications of mutation operator, the result becomes feasible because the gene is retrieved from the machine pool when the constraint-chromosome approach is used.

The best solutions for the 10 problem sets are given in Table 5. The number of part types allocated and the number of part types that are not allocated are shown, along with the values of the respective chromosomes and number of generations required to converge. As an example, from data set 1, from eight part types that had to be allocated, five part types (part types 6, 5, 7, 3 and 1) are allocated, whereas the other three part types (part types 4, 8 and 2) are not allocated. On the basis of the value of the respective chromosomes, the details of the allocation for the part types are given in Table 6.

The comparative study performed between the proposed algorithm and existing literature is given in Table 7. The proposed algorithm of the current study is compared with the CCGA with heuristics developed by Shanker and Srinivasulu [1], Mukhopadhyay et al. [2], Nagarjuna et al. [18] and Yogeswaran et al. [41]. Based on the results, the proposed algorithm performs better than most heuristics available in literature in terms of the minimization of system unbalance as well as COFs (Table 7). The proposed algorithm achieves the best solutions for 6 of 10 data sets tested. The average COF (1.8045) in the

TABLE 6. Detailed allocation for data set 1

Part Type	Operation	Machine Number
6	1	4
	2	2
	3	2
5	1	2
	2	2
7	1	2
	2	1
3	1	1
	2	3
1	1	3

TABLE 7. Comparison of the proposed CCGA with other heuristics

Data Set	Number of Part Types	Total Number of Operations	[15]		[10]		[18]		[41]		Proposed Algorithm	
			TH	SU	TH	SU	TH	SU	TH	SU	TH	SU
			COF		COF		COF		COF		COF	
1	8	19	39	253	42	122	48	14	48	14	52	0
			1.3557		1.4615		1.4615		1.5927		**1.6500**	
2	6	9	51	388	63	202	46	18	63	22	64	15
			1.4965		1.7578		1.6208		1.8516		**1.8689**	
3	5	9	63	288	79	286	69	94	73	28	73	28
			1.6475		1.851		1.8245		1.9095		1.9095	
4	5	10	51	819	51	819	51	819	51	819	51	819
			1.5734		1.5734		1.5734		1.5734		1.5734	
5	6	10	62	467	76	364	53	175	61	264	61	69
			1.5726		1.8104		1.6062		1.6651		1.7667	
6	6	8	51	548	62	365	64	69	64	37	64	7
			1.4132		1.6592		1.8408		1.8731		1.8731	
7	6	10	54	189	66	147	54	165	63	231	63	21
			1.5939		1.7696		1.6064		1.6874		**1.7968**	
8	7	15	36	459	36	459	44	13	48	63	54	56
			1.2752		1.2752		1.6218		1.6529		**1.7423**	
9	7	12	79	462	88	309	88	309	88	309	88	56
			1.6571		1.8391		1.8391		1.8391		**1.9708**	
10	6	16	44	518	56	320	54	82	56	122	67	205
			1.3869		1.6692		1.7633		1.7723		**1.8932**	
Average COF			1.4972		1.6666		1.6889		1.7417		**1.8045**	

Abbreviations: TH, throughput; SU, system unbalance.

last row of Table 7 clearly shows that the proposed CCGA is superior in performance over the other heuristics considered for evaluation. In addition, most of the superior results are achieved in the data sets with high number of operations (data sets 1, 7, 8, 9 and 10). These figures demonstrated that the proposed algorithm in the current study is capable of handling high-operation machine-loading problems, such as those that usually occurred in actual practice.

In addition, all data sets achieve fast convergence with good results. These figures demonstrate the effectiveness of the chromosome representation as well as the crossover and mutation operators that capably searches diversity for the new solution. Furthermore, the arrangement of gene formation that carries the relevant information required by the fitness calculation contributes to the faster processing.

6. **Conclusion.** The present work discusses the machine-loading problem as a part of the very important aspect of planning horizon. It deals with the challenge of locating

the available resources (machine) to load the part types, while considering the constraints of the problem, such as the machine time available, the number of operations, and the allowable machines that may be allocated for the operations.

The main contribution of the present research is an efficient heuristic approach based on CCGA, which can solve a machine-loading problem of FMS. Whereas some of the previous studies considered part type sequence and operation allocation as separate but interconnected components, the present research adopts an approach of treating part type sequencing and machine allocation problem as one main goal. Based on such a goal, the tasks are carried out concurrently in this work. This process is repeated iteratively until a termination condition is met, after which the optimal or near-optimal solution has been obtained.

Exhaustive computations are been carried out to assess the effectiveness of the proposed algorithm, and its performance has been compared with the previous studies. From the results, the proposed CCGA offers better results for most of the test problems, in which the COF increased by 3.60% compared with the best result of the other heuristics. The right chromosome representation to map the machine-loading problem, which considers constraints as well as the efficient genetic operators, leads to the good result. In addition, the proposed algorithm achieves fast convergence, such that most of the best results are obtained at an early number of generations. This ability is very important for the manufacturing industry that has always prioritized the use of various methods, by which to save processing time and cost in generating machine allocation planning. The results also show that most of the superior results occur on the data sets containing high number of operations, thus demonstrating the ability of the proposed algorithm to effectively and efficiently handle high-number operation machine-loading problems that occur in actual manufacturing scenarios.

The proposed approach can be applied to similar constraint optimization problems, particularly in allocation and scheduling, where optimizing an objective function is subjected to resource and constraints. Nevertheless, the proposed algorithm is tested on the data sets available in literature, and the extension of this work may be tested on actual large-scale problems. Additionally, this work covers only certain constraints with the assumption of sufficient resources available, such as fixtures, jigs and pallets, in the shop floor. The work may be extended further to cover these resources as part of additional constraints. Finally, the proposed algorithm can be extended to solve multi-objective machine-loading problems and offer more flexible attributes.

Acknowledgment. The main author wishes to thank Universiti Sains Malaysia for the support it has extended in the completion of the present research through the University Short Term Grant No: 304/PKOMP/639009. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers.

REFERENCES

[1] N. Viswanadham and Y. Narahari, *Performance Modeling of Automated Manufacturing Systems*, Prentice-Hall, Inc., India, 1992.

[2] K. E. Stecke, Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems, *Management Science*, vol.29, no.3, pp.273-288, 1983.

[3] M. Berrada and K. E. Stecke, A branch and bound approach for machine load balancing in flexible manufacturing systems, *Management Science*, vol.32, no.10, pp.1316-1335, 1986.

[4] S. Hwang, *Models for Production Planning in Flexible Manufacturing System*, Thesis, University of California, 1986.

[5] M. Liang and S. P. Dutta, An integrated approach to the part selection and machine loading problem in a class of flexible manufacturing systems, *European Journal of Operational Research*, vol.67, no.3, pp.387-404, 1993.
[6] A. Kumar, Prakash, M. K. Tiwari, R. Shankar and A. Baveja, Solving machine-loading problem of a flexible manufacturing system with constraint-based genetic algorithm, *European Journal of Operational Research*, vol.175, no.2, pp.1043-1069, 2006.
[7] Srinivas, M. K. Tiwari and V. Allada, Solving the machine-loading problem in a flexible manufacturing system using a combinatorial auction-based approach, *International Journal of Production Research*, vol.42, no.9, pp.1879-1893, 2004.
[8] J. C. Ammons, C. B. Lofgren and L. F. McGinnis, A large scale machine loading problem in flexible assembly, *Annals of Operations Research*, vol.3, no.7, pp.317-332, 1985.
[9] K. Shanker and Y.-J. J. Tzen, A loading and dispatching problem in a random flexible manufacturing system, *International Journal of Production Research*, vol.23, no.3, pp.579-595, 1985.
[10] 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.2213-2228, 1992.
[11] M. K. Tiwari, B. Hazarika, N. K. Vidyarthi, P. Jaggi and S. K. Mukhopadhyay, A heuristic solution approach to the machine loading problem of an FMS and its petri net model, *International Journal of Production Research*, vol.35, no.8, pp.2269-2284, 1997.
[12] S. Sarin and C. Chen, The machine loading and tool allocation problem in a flexible manufacturing system, *International Journal of Production Research*, vol.25, no.7, pp.1081-1094, 1987.
[13] G. K. Nayak and D. Acharya, Part type selection, machine loading and part type volume determination problems in FMS planning, *International Journal of Production Research*, vol.36, no.7, pp.1801-1824, 1998.
[14] S. K. Mukhopadhyay, M. K. Singh and R. Srivastava, FMS machine loading: A simulated annealing approach, *International Journal of Production Research*, vol.36, no.6, pp.1529-1547, 1998.
[15] K. Shanker and A. Srinivasulu, Some solution methodologies for loading problems in a flexible manufacturing system, *International Journal of Production Research*, vol.27, no.6, pp.1019-1034, 1989.
[16] U. M. B. S. Sarma, S. Kant, R. Rai and M. K. Tiwari, Modelling the machine loading problem of FMSs and its solution using a tabu-search-based heuristic, *International Journal of Computer Integrated Manufacturing*, vol.15, no.4, pp.285-295, 2002.
[17] 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.
[18] N. Nagarjuna, O. Mahesh and K. Rajagopal, A heuristic based on multi-stage programming approach for machine-loading problem in a flexible manufacturing system, *Robotics and Computer-Integrated Manufacturing*, vol.22, no.4, pp.342-352, 2006.
[19] Y. Li, W. H. Ip and D. W. Wang, Genetic algorithm approach to earliness and tardiness production scheduling and planning problem, *International Journal of Production Economics*, vol.54, pp.65-76, 1998.
[20] M. K. Tiwari and N. K. Vidyarthi, Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach, *International Journal of Production Research*, vol.38, no.14, pp.3357-3384, 2000.
[21] W. H. Ip, Y. Li, K. F. Man and K. S. Tang, Multi-product planning and scheduling using genetic algorithm approach, *Comput. Ind. Eng.*, vol.38, pp.283-296, 2000.
[22] P. Pongcharoen, C. Hicks and P. M. Braiden, The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure, *European Journal of Operational Research*, vol.152, no.1, pp.215-225, 2004.
[23] F.-T. Lin and T.-R. Tsai, A two-stage genetic algorithm for solving the transportation problem with fuzzy demands and fuzzy supplies, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.4775-4785, 2009.
[24] Y. Maeda and Q. Li, Parallel genetic algorithm with adaptive genetic parameters tuned by fuzzy reasoning, *International Journal of Innovative Computing, Information and Control*, vol.1, no.1, pp.95-107, 2005.
[25] F.-T. Lin, Simulating fuzzy numbers for solving fuzzy equations with constraints using genetic algorithms, *International Journal of Innovative Computing, Information and Control*, vol.6, no.1, pp.239-253, 2010.

[26] K. Okuhara, J. Shibata and H. Ishii, Adaptive worker's arrangement and workload control for project management by genetic algorithm, *International Journal of Innovative Computing, Information and Control*, vol.3, no.1, pp.175-188, 2007.

[27] H. S. Kim and D. H. Kim, Genetic algorithm-based dynamic channel allocation to minimize the inter-cell interference in downlink wireless communication systems, *International Journal of Innovative Computing, Information and Control*, vol.6, no.11, pp.5179-5190, 2010.

[28] H. Suryoatmojo, A. A. Elbaset, Syafaruddin and T. Hiyama, Genetic algorithm based optimal sizing of PV-diesel-battery system considering CO_2 emission and reliability, *International Journal of Innovative Computing, Information and Control*, vol.6, no.10, pp.4631-4649, 2010.

[29] T.-H. Tan, Y.-F. Huang and F.-T. Liu, Multi-user detection in DS-CDMA systems using a genetic algorithm with redundancy saving strategy, *International Journal of Innovative Computing, Information and Control*, vol.6, no.8, pp.3347-3364, 2010.

[30] Z.-H. Che, Using hybrid genetic algorithms for multi-period product configuration change planning, *International Journal of Innovative Computing, Information and Control*, vol.6, no.6, pp.2761-2785, 2010.

[31] S. Ono, Y. Hirotani and S. Nakayama, A memetic algorithm for robust optimal solution search – Hybridization of multi-objective genetic algorithm and Quasi-Newton method, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.5011-5019, 2009.

[32] S.-H. Chen, P.-C. Chang, Q. Zhang and C.-B. Wang, A guided memetic algorithm with probabilistic models, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.4753-4764, 2009.

[33] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., 1989.

[34] R. Saravanan, *Manufacturing Optimization through Intelligent Techniques*, CRC Press, New York, 2006.

[35] M. W. Bloomfield, J. E. Herencia and P. M. Weaver, Analysis and benchmarking of meta-heuristic techniques for lay-up optimization, *Computers & Structures*, vol.88, no.5-6, pp.272-282, 2010.

[36] Z. Michalewicz and G. Nazhiyath, Genocop iii: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints, *IEEE International Conference on Evolutionary Computation*, vol.2, pp.647-651, 1995.

[37] D. Cormier, P. O'Grady and E. Sanii, A constraint-based genetic algorithm for concurrent engineering, *International Journal of Production Research*, vol.36, no.6, pp.1679-1697, 1998.

[38] U. K. Yusof and S. Deris, Constraint-based genetic algorithms for machine requirement of semiconductor assembly industry: A proposed framework, *The 3rd Asia International Conference on Modelling & Simulation*, Bandung, Indonesia, pp.29-34, 2009.

[39] C.-F. Tsai and K.-M. Chao, An effective chromosome representation for optimising product quality, *The 11th International Conference on Computer Supported Cooperative Work in Design*, pp.1032-1037, 2007.

[40] N. B. Ho and J. C. Tay, Genace: An efficient cultural algorithm for solving the flexible job-shop problem, *Congress on Evolutionary Computation*, vol.2, pp.1759-1766, 2004.

[41] 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.