International Journal of Innovative
Computing, Information and Control
Volume **10**, Number **6**, December **2014**

ICIC International ©2014 ISSN 1349-4198
pp. 2325–2344

A HYBRID OF BIO-INSPIRED AND MUSICAL-HARMONY APPROACH FOR MACHINE LOADING OPTIMIZATION IN FLEXIBLE MANUFACTURING SYSTEM

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

[1]School of Computer Sciences
Universiti Sains Malaysia
11800 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 2013; revised April 2014

Abstract. *Manufacturing industries are facing fierce challenges in handling product competitiveness, shorter product cycle time and product varieties. The situation demands a need to improve the effectiveness and efficiency of capacity planning and resource optimization while still maintaining their flexibilities. Machine loading – one of the important components of capacity planning is known for its complexity that encompasses various types of flexibilities pertaining to part selection, machine and operation assignment along with constraints. Various studies are done to balance the productivity and flexibility in flexible manufacturing system (FMS). From the literature, researchers have developed many approaches to reach a suitable balance of exploration (global improvement) and exploitation (local improvement). We adopt a hybrid of population approaches; hybrid constraint-chromosome genetic algorithm and harmony search algorithm (H-CCGaHs), to solve this problem that aims at mapping a feasible solution to the domain problem. The objectives are to minimize the system unbalance as well as to increase the throughput while satisfying the constraints such as machine time availability and tool slots. The proposed algorithm is tested for its performance on 10 sample problems available in FMS literature and compared with existing solution approaches.*

Keywords: Flexible manufacturing system, Machine loading, System unbalance, Throughput, Hybrid genetic algorithm and harmony search

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

One of the main purpose of FMS is to achieve efficiency of a well-balanced transfer line while retaining the flexibility of the job shop [2]. This includes the efficiency of

handling machine loading problems. The machine loading problem – one of the important aspects of capacity planning in FMS refers to assigning the machines, operations, and the necessary tools for selected part types to perform these operations within the technological constraints in order to maximize throughput and minimize system unbalance [2, 3].

In the FMS environment, machine loading problem dwells on how the part types should be assigned together with the allocation of the tools so that the productivity is optimized [3]. This is achieved given a set of part types to be produced with a set of tools that are needed for processing the part types on a set of machine, together with using a set of resources such as material handling appliances, pallets and fixtures. The function is able to minimize the idle time of the machine, leading to maximization of machine utilization and enhancing overall system output.

The machine loading problem optimization has been studied by many researchers over many years as the area of study is receptive to new approaches that have proven to achieve better results. Furthermore, the machine loading problem is only a fraction of the real manufacturing problem, where each proposed work should be looked into with a possibility of applying it onto the real manufacturing environment. From the literature, it is concluded that the problem pertaining to the machine loading covers many objectives [2, 4]. Due to many objectives covered which involves many simultaneous determinations of many factors such as throughput, sequence and workload balancing, machine loading problem lies under the broad category of NP-hard problem [5] where the requirement is to satisfy various technological constraints. Shanker and Tzen [6] worked on balancing machine workload and meeting the part types due date, while Ammons et al. [4] resolved the loading problem to meet a bi-criteria objectives. Later, Mukhopadhyay et al. [7] and Tiwari and Vidyarthi [8] solved machine loading problem with an objective of maximizing throughput and minimizing system unbalance using heuristic approaches.

Generally, machine loading problem can be handled using two main approaches: (i) optimization-based and (ii) heuristic-based. Optimization-based methods are robust in their applicability, though they tend to become impractical when the problem size increases which usually occurs in manufacturing scenarios, whereas heuristic approaches are usually dependent on rules and constraints of individual problem. Because of this nature, heuristic-based approaches always face difficulty in estimating results in a changed problem or environment, which poses issues in handling the ever-changing manufacturing requirements.

These limitations motivate the researchers to enhance the method and to look for innovative searching technique to further improve heuristic-based approaches. One of the popular approaches is genetic algorithm (GA) which is known for its capability in intelligent probabilistic searching. Tiwari and Vidyarthi [8] used GA to allocate resources in ordering and increasing equipment utilization and throughput while Li et al. [9] proposed GA to handle problems with multi-period and multi-level capabilities balancing issues. Kumar et al. [3] proposed constraint-chromosome GA in handling complex constraints in the FMS loading problem.

GA has proven to be robust in handling many types of manufacturing optimization problems. For example, Saravanan [10] came out with many extensive examples and comparisons of AI-based techniques (i.e., genetic algorithms, tabu search, simulated annealing, particle swarm optimization and ant colony optimization) that are applied to handle several manufacturing optimization problems. In summary, GA is able to solve all these problems successfully, whereas other techniques are able to solve certain types of problems only. Nevertheless, GA may have the tendency to converge towards local optima and often lose out in finding the global optimum of the problem [11]. In other

words, it has the tendency of focusing on the local optimum rather than exploring new solution to achieve global optimum.

Integrating GA into other algorithms or approaches has also been proposed in previous studies. Che [12] used hybrid GA to enhance productivity while considering product configuration change. Ono et al. [13] 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. [14] proposed a guided mimetic algorithm to solve scheduling problems and Yogeswaran et al. [15] solved machine loading problem in FMS using GA and simulated annealing algorithm.

Another potential algorithm that is able to handle machine loading problem effectively is harmony search (HS). It is a new metaheuristic population-based algorithm created by Geem et al. [16]. This heuristic algorithm is derived from an artificial phenomenon found in musical performance that always seeks for a better harmony. HS has been applied to several computational optimization problems such as school bus routing [17], music composition [18] and timetabling problem [19]. To the best knowledge of the author, none of the work has focused on machine loading problem.

Geem et al. [16] stated that the main features of HS that make it different from other methods are: HS makes a new vector after considering all existing vectors rather than considering only two (parents) as in GA, and HS does not require the setting of initial values of decision variables. These features increase the flexibility in order to find better solutions. The study also showed that HS is capable of achieving faster convergence compared with GA and simulated annealing. In addition, Mukhopadhyay et al. [20]'s study concluded that HS poses a strong explorative power, which is a very important characteristic for evolutionary algorithms (EA).

Integrating HS into other algorithms or approaches has also been proposed in previous studies. Alia et al. [21] worked on clustering algorithm based on the harmony search (HS) hybridized with fuzzy c-means called DCHS to automatically segment the brain MRI image in an intelligent manner, Wang et al. [22] proposed a hybrid modified global-best harmony search (hmgHS) algorithm for solving the blocking permutation flow shop scheduling problem with the makespan criterion, while Li and Wang [23] proposed a global optimization by merging the mechanisms of harmony search (HS) and differential evolution (DE).

Based on the above literatures, the hybrid of the algorithms may increase the ability to produce better results as opposed to using a single algorithm in solving the problem. In addition, in combining the strength of GA (exploitative) and HS (explorative), the hybrid of those two algorithms may have potential of producing better results than an individual algorithm is able to. In our study, we hybridize two algorithms, GA and HS to create a more robust and effective algorithm in finding a better result. 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 which promotes higher machine utilization. In addition, machine loading policy aims for the maximization of total system output which is interpreted as throughput.

The remainder of the paper is organized as follows. Section 2 describes the problem and model formulation. Section 3 discusses on the solution proposed to solve the problem. Section 4 discusses on the results of the proposed solution. Finally, Section 5 concludes the paper.

2. **Problem Description and Model Formulation.**

Part type selection and machine loading arrangement constitute two major components of the strategic planning problem of FMS. Part type selection deals with selecting a set of part types to be manufactured

during the upcoming planning horizon whereas the loading problem concerns about the allocation of operations and the required machine and tools for the selected part types. Most of the earlier researchers have addressed these problems separately due to their complexities; hence the solution of part type selection problem may lead to infeasible result for real-life loading problem.

Part types selection may be performed by two types of operations: essential operation and optional operation. Essential operation is the operation that can be performed only on a specific machine using a certain number of tool slots whereas optional operation implies that it can be performed on a number of machines with the same or varying processing time and tool slots. Since essential operation is required to be performed at particular machine/s, the flexibility lies in the selection of machines that are set for optional operation where there are more opportunities in improving the machine allocation which may produce better results.

As previously mentioned, the objective functions for this research are the maximization of throughput and minimization of system unbalance combined to attain the overall function for machine loading problem. Based on the implication, it is very important to have both objective functions combined in a logical manner that will also consider the technical constraints.

2.1. **Notations, formulation of objective function and constraints.** The subscript and parameter notations used to demonstrate the objective functions are:

Subscripts

N	Total number of parts
M	Total number of machines
H	Planning horizon (480 minutes)
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

UT_m	under-utilized time on machine m
OT_m	over-utilized 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^r_{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

2.2. **Objective functions.**

- The first objective of this research is to minimize the system unbalance leading to maximizing system utilization that is formulated as

$$F_1 = \frac{M * H - \sum_{m=1}^{M} UT_m - OT_m}{M * H} \tag{1}$$

- The second objective function is to maximize throughput, leading to maximization of system efficiency that is formulated as

$$F_2 = \frac{\sum_{i=1}^{N} \beta_i * \alpha_i}{\sum_{i=1}^{N} \beta_i} \tag{2}$$

  Both objectives have values between zero and one. A perfectly balanced work-allocation will have $F_1 = 1$, and full completion of all the considered products within the considered time frame give $F_2 = 1$. Therefore, the objective function strives to reach $F_1 = 1$, and $F_2 = 1$, that is, to maximize both F_1 and F_2. For making the objective function more robust, the objective is defined as a weighted average of these two objectives.
- The third objective function is a combination of both of the above two objective functions and is formulated as

$$F = Maximize \left(\frac{W_1(F_1) + W_2(F_2)}{W_1 + W_2} \right) \tag{3}$$

  where W_1 and W_2 are the corresponding weight associated with the system unbalance and throughput respectively. 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:
System unbalance:

$$\sum_{m=1}^{M} UT_m - OT_m \geq 0. \tag{4}$$

Non-splitting of part type:

$$\sum_{m=1}^{M} \sum_{j=1}^{J} \alpha_{imj} = \alpha_i * J, \quad i = 1, 2, \ldots, N. \tag{5}$$

Available time on machines:

$$\sum_{j=1}^{P} t_{imj} \leq t_{im}^{a}, \quad i = 1, 2, \ldots, N. \tag{6}$$

Unique part type routing:

$$\sum_{mY_{ij}} \alpha_{imj} \leq 1, \quad i = 1, 2, \ldots, N; j = 1, 2, \ldots, J. \tag{7}$$

Number of machines and machine remaining time:

$$T_{imj}\alpha_{imj} \geq 0, \quad i = 1, 2, \ldots, N; m = 1, 2, \ldots, M. \tag{8}$$

$$t_{imj}\alpha_{imj} \geq 0, \quad i = 1, 2, \ldots, N; m = 1, 2, \ldots, M. \tag{9}$$

The integrity of decision variables:

$$\alpha_i = \begin{cases} 1 & \text{if part type } i \text{ is selected} \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

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

$$\alpha_{im} = \begin{cases} 1 & \text{if part type } i \text{ is assigned to machine } m \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

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

$$\alpha_{imj} = \begin{cases} 1 & \text{if operation } j \text{ of part type } i \text{ is assigned to machine } m \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

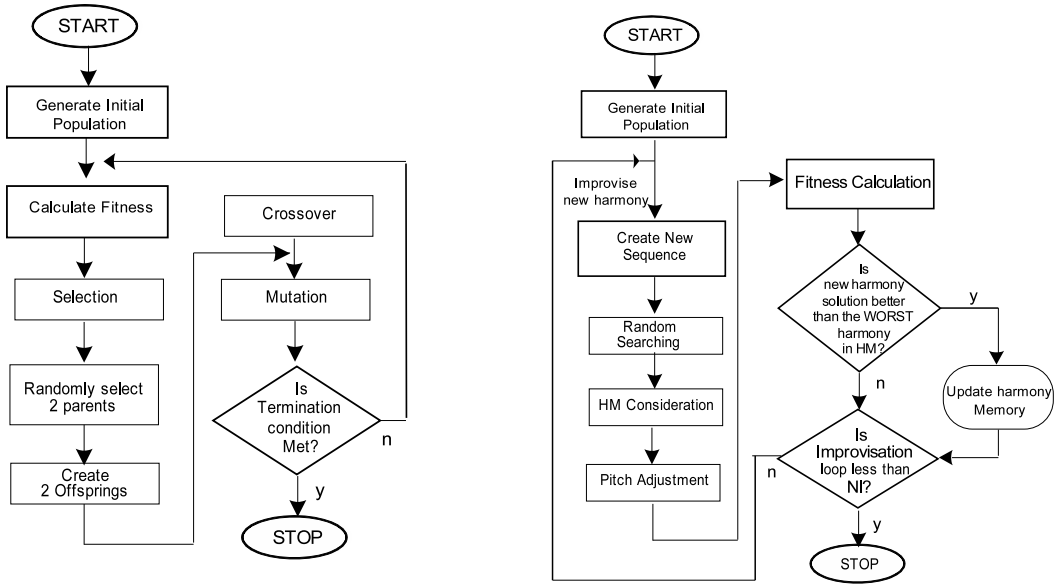
$$i = 1, 2, \ldots, N; m = 1, 2, \ldots, M; j = 1, 2, \ldots, J$$

In the above constraints, constraint (4) defines that the sum of the idle time remaining on machines after allocation of all feasible part types must be either zero (100% utilization of system) or a positive value. Constraint (5) ensures 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. Constraint (6) forces the available time on machine to be greater than or equal to the time required by the next part type to be assigned to this machine. Constraint (7) ensures that the operation must be completed on the same machine and expressed once a machine is selected. Constraints (8) and (9) dictate that 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. Constraints (10), (11) and (12) ensure the integrity of decision variables where the decision variables possess a value of 0 and 1 integers.

3. Solution by a Hybrid of Constraint-Chromosome Genetic Algorithm and Harmony Search.

3.1. **Genetic algorithm.** Genetic Algorithm (GA) is a population-based method that is inspired by the principle of natural evolution [24]. It is motivated by the mechanism of natural selection, which is a biological process that does the selection based on 'the survival of the fittest'. GA steps involve initialization, where the initial populations are generated using a random function. These populations are assigned a fitness value based on the objective function. In order to generate the possibility for more solutions, new populations are created based on the selection criteria. Genetic operators; crossover and mutation are applied to these newly-formed populations and the objective function is calculated. These steps (the formation of new solution until fitness calculation) will be repeated until the satisfactory result is achieved. Figure 1(a) shows the basic GA process.

3.2. **Harmony search.** Harmony search (HS) mimics music harmony that produces an aesthetic and pleasing combination of sounds. Musical performances seek for the best performance (fantastic harmony) which is determined by aesthetic estimation, similar to the optimization algorithms that search for the best performance (global optimum, maximum benefit or efficiency) which is determined by an objective function evaluation. Since aesthetic estimation is established by a set of harmonic sounds played by instrument combinations, the objective function evaluation is determined by the set of values produced by component variables. Better aesthetic estimation can be improved via practice, the value of objective functions can be improved via iterations. Figure 1(b) shows the basic HS process.



(a) Basic GA process (b) Basic HS process

FIGURE 1. Basic process

3.3. **Proposed hybrid constraint-chromosome genetic algorithm and HS algorithm (H-CCGaHs).** Even though GA is known as a global search technique, the effectiveness of its usage highly depends on how well it can handle problem constraints. Due to that, there are a number of techniques that have been developed to handle constraints. Michalewicz and Nazhiyath [25] surveyed several constraint methods for GA where the most common approach is to add penalty function to the objective function in order to transform a non-constraint problem into a problem with constraint. In this penalty function, the basic rule is to add zero penalty when all constrains are satisfied, and non-zero penalty otherwise. However, quantifying constraint violation is not always practical especially when we are dealing with non-numeric variables. Therefore, applying penalty functions directly inside the FMS framework can be a tedious task. Cormier et al. [26] proposed constraint-based genetic algorithm to solve problems in concurrent engineering. This concept is adopted in our current work to solve FMS problem, with some modifications.

For this reason, we put an effort to find an efficient coding of each individual with respect of taking all the constraints of machine loading problem. In our work, we consider constraints during the initialization of population which we call constraint-chromosome genetic algorithm (CCGA). The chromosome is defined as a part type sequence of the problem where the sequence will determine the arrangement of the processing. The part types are arranged based on random selection from the scheduled part types, where each gene is checked against its existence to ensure that no same gene is generated in the chromosome. Each part type is assigned a number of operations required to be performed with respective available machines. To ensure the feasibility of the solution, each gene in part type sequence will retrieve the respective operation that is assigned to it. Since the operation sequence for each part type has to follow the sequence, which means the first operation need to be allocated first before the second one, the sequence must be maintained. A sample of part sequence for Table 1 (dataset 8) is shown in Figure 2.

TABLE 1. Description of the dataset 8 (adopted from [27])

Part Type	Batch Size	Number of Operation	Operation No	Machine No	Unit Proc Time (min)	Tool Slot	Total Proc Time
1	9	2	1	2,3	22	2	198
			2	3	25	1	225
2	8	1	1	3	20	1	160
3	9	3	1	1,4	25	1	225
			2	4	25	1	225
			3	2	22	1	198
4	7	2	1	3	24	1	168
			2	4	19	1	133
5	14	2	1	4,1	26	2	364
			2	3	11	3	154
6	13	3	1	1,2,3	25	1	325
			2	2,1	17	1	221
			3	1	24	3	312
7	10	2	1	4	16	1	160
			2	4,2,3	7	1	70

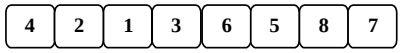


FIGURE 2. A sample of part sequence

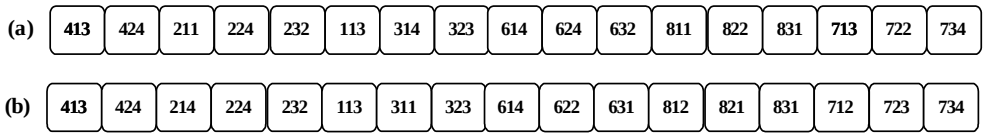


FIGURE 3. (a) Possible value of part-operation-chromosome, (b) another possible value of part-operation-chromosome

The convergence speed of a genetic algorithm is very important 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, each unit is assigned a combination number of part type, operation and machine. The feasible machine for each operation will be randomly selected based on the range of machines that should be performed for that particular part type and operation. For example, part type 4 in part type sequence [4 2 1 3 6 5 7] is considered as an essential operation, whereby the operation can be performed on a particular machine only, and allows machine 3 to be assigned to operation 1. Since both the operation 1 and 2 of part type 4 is considered as an essential operation, there is no change in the gene that generated the part-operation chromosomes, as shown in Figures 3(a) and 3(b). On the other hand, operation of part type 6 is an optional operation where this operation can be 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, 2 or 3 where the total processing time for both varies which can lead to a different result.

As previously mentioned, the strength of GA lies in its ability to develop a robust strategy in handling the manufacturing optimization problems with constraints; while HS offers flexibility in finding a better solution as well as possesses the ability to converge faster compared to other methods. In other words, GA is known for its exploitation ability to achieve a good solution, while HS has an ability to explore more solutions. These two characteristics are the main characteristics of EA that offer a good balance in solving the real machine loading problem. Considering the strength of these algorithms, it is logical to hybridize these two algorithms as well as to impose the constraints in population creation to achieve a better result. This study proposes the hybrid of CCGA and HS called H-CCGaHs.

This model uses GA as a base for population initialization. The population is then divided into two equally-sized populations that perform GA and HS operators separately. The success of GA and HS is very much dependent on how efficient the respective operators are in searching for a better solution. Due to this, H-CCGaHs is proposed in such a way that the populations will interchangeably operate a different set of operators for each generation/iteration. The population swap between CCGA and HS may effectively avoid being trapped in a local optima as well as to ensure robustness (e.g., GA population will be used for HS next generation processing and vice versa). In the HS process, this population is referred to as harmony memory (HM). The structure of the proposed model is described in Figure 4.

3.4. **Parameters initialization.** Based on Figure 4, the parameter initialization for GA and HS are being set. For GA, the number of generation (NG), the size of population (SP), the crossover rate (CR) and the mutation rate (MR) are initialized with selected values. NG determines the iteration set for the termination of the algorithm, SP is the size of the feasible solutions. CR is used to vary the programming of a chromosome from one generation to the next, while MR is to maintain genetic diversity of one generation of a population to the next. For HS, the parameters are the harmony memory consideration rate (HMCR), the pitch adjustment rate (PAR) and the harmony memory size (HMS). HMCR is similar to the crossover rate in genetic algorithm, while PAR behaves similar to MR in determining the number of part types to be moved to another position or swapping them with other part types. HMS is the size of the solution matrix that is similar to SP.

3.5. **Population initialization.** We use GA approach for population initialization where each population consists of a chromosome that is represented by a string. The selection of a string format for the individual is the first and a very important step to a successful implementation of genetic algorithm. Initial GA populations are generated using a random function, where the sequence of part type is randomly generated from the part type scheduled for the processing. The sequence determines the arrangement of the part type processing, and as part of the constraint, the same part type cannot exist twice in the chromosome string.

3.6. **Fitness evaluation.** After chromosomes 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, since the chromosomes selected for mating are based on their fitness. The objective function will decide whether part type sequence with selected machine is good for the machine loading problem. Machine loading problem in FMS involves multiple objective functions, so the model should be robust for the user to specify several objectives. For our problem, the objective function will determine the throughput (TH) and system unbalance (SU). Throughput is defined as units of part types produced for all selected part types during the planning horizon where the units

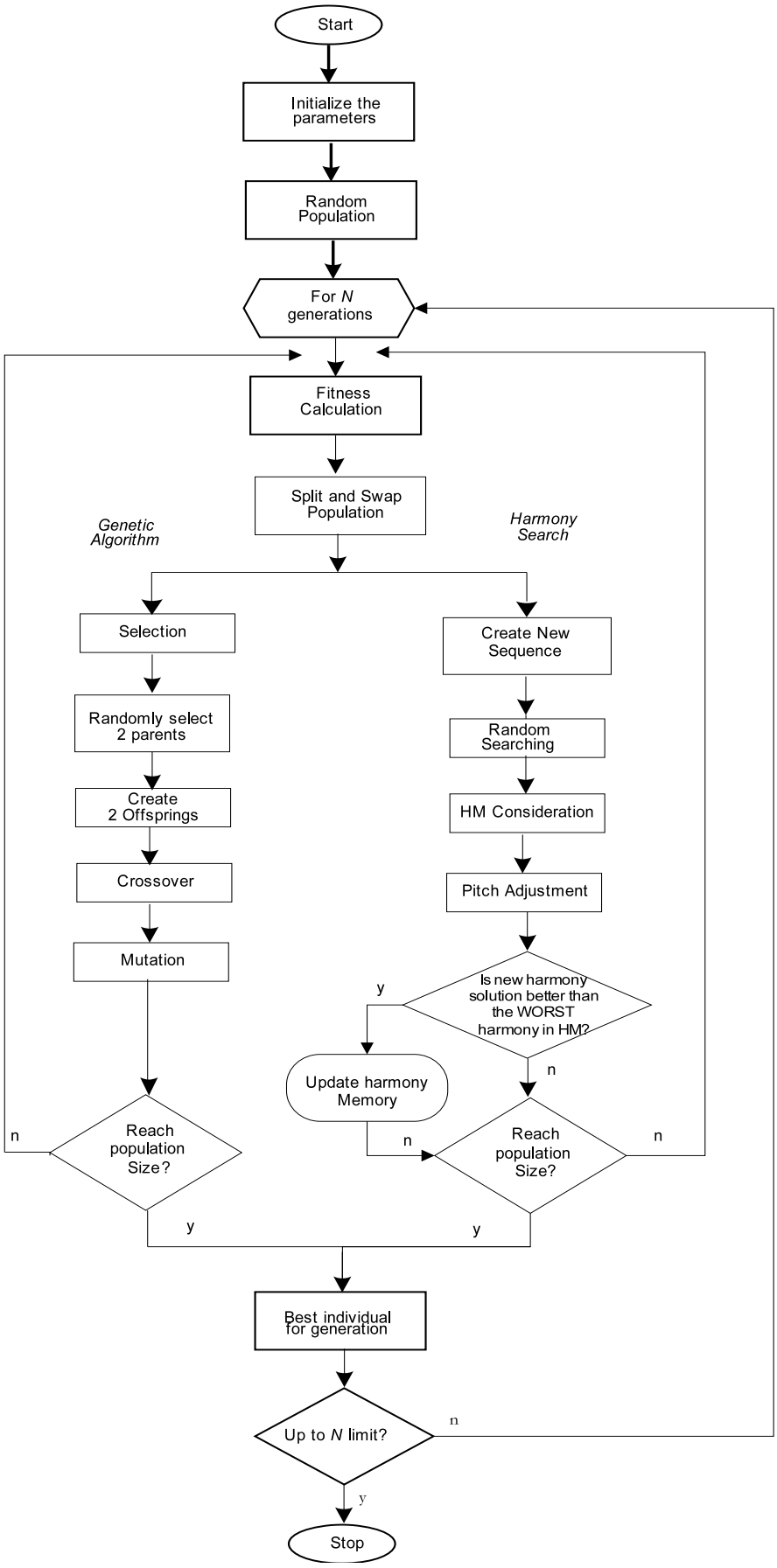


FIGURE 4. Flow chart of the proposed H-CCGaHs model

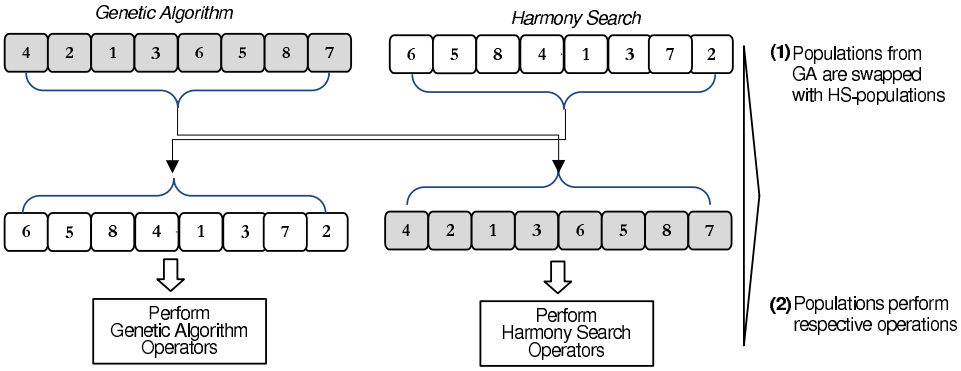


FIGURE 5. Population swapping between GA and HS

are the sum of batch sizes. System unbalance refers to sum of the over-utilized (OT_m) or under-utilized (UT_m) time on all machines after the allocation of all part types.

Since the chromosomes are constructed from valid genes, the result is also expected to be a feasible result. The chromosome carries all relevant information to calculate the fitness value including the operations required for each part type and the machine which the operation is to be performed. From this information, we can determine the total processing time and machine slots for each chromosome and are able to calculate the fitness value and determine which part type in the string can be assigned.

3.7. **Population splitting and swapping.** After the fitness calculation, the population is divided into half where these populations will perform GA and HS operators respectively. Although the original HS does not consider the population size to determine the processing limit criteria, HS algorithm is improvised to include population size as part of its process in order to accommodate GA approach. Furthermore, this process may also enhance HS capability of selecting a better solution. To create diversity for next generations, these populations are swapped interchangeably (i.e., GA populations perform HS operators and vice versa). Through swapping the populations generated by the two algorithms and running against the other algorithm's operators, it will create more chances of discovery of unvisited solutions and avoids creating repeated or a similar solution. A sample of populations swapping is shown in Figure 5.

3.8. **Genetic algorithm operators.**

3.8.1. *Selection.* Selection process provides the driving force in genetic algorithm to determine the continuation of evolutionary process flow. Previous researches have proposed, tested and compared many selection methods. This research proposes the use of simple roulette wheel selection method where the selection will pick the parents for the next mating based on the fitness proportionate selection. In this proportionate selection, the objective function will assign a fitness value to each chromosome to associate a probability of selection with each individual chromosome, so the higher the fitness, the better the chance of being selected.

3.8.2. *Crossover and mutation operators.* The chromosome has to be broken up to give the result that is legal to our problem. Our chromosome also falls into the category where the same gene cannot appear more than once and direct swap between parents will tend to create duplication or deletion of certain genes; hence creating infeasible result. Due to that, we apply ordered chromosomes with two-point crossover. Two crossover points of the first parent will be randomly generated to determine the first portion of the first offspring. The remaining genes of chromosome in the first offspring will be taken from

the second parent, considering the same gene is not to be taken again. The information after the crossover point is ordered the same as that in the other parent. Second offspring will repeat the same process and starts creating the second offspring with second parent.

Mutation avoids the potential of being trapped in local minima by preventing the population of chromosomes from becoming too similar to each other. This phenomena will eventually slow down the evolution process and prevent the ability of the algorithm to achieve good results. For mutation, we use reciprocal exchange mutation, where two randomly selected genes are being swapped in the same chromosome. This operation promotes diversity where it opens possibility for a new searching space. 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 make the selection random (or semi-random) with weighing factors towards the fitter ones.

3.9. **Harmony search algorithm operators.**

3.9.1. *Memory considerations, random searching and pitch adjustments operators.* A new feasible harmony solution is generated based on three operators – memory considerations, random searching and pitch adjustments.

With the probability of HMCR, memory consideration operator selects the part type positions of a new harmony solution based on the solutions stored in HM. HMCR has a probability of selecting historical positions from feasible machine loading stored in HM. As an example, if HMCR $= 0.90$, it indicates the number of part types to be selected randomly from a historical positions in the HM to be 90%. Based on (1-HMCR) probability, part types will be assigned based on the random positions from the whole available position. This operator is equivalent to the mutation operator in genetic algorithm [16] to ensure that algorithm avoids the potential of being trapped in local minima by preventing the population of matrix from becoming too similar to each other.

To improvise the solutions as well as to escape local optima, another option is introduced that mimics the pitch adjustment of each instrument for tuning the ensemble. Based on probability of PAR, the operator will select the part type depending on memory consideration rather than random consideration. The obtained part type moves to neighbouring part type with the probability value of (PAR $\times$ HMCR) with the value of $0 \leq PAR \leq 1$. In case of machine loading, the selected part type will move to a valid position with the probability of (PAR $\times$ HMCR), where the pitch adjustment operator works similar to the neighbourhood structures of local based heuristics that are concerned with the exploitation of the new harmony solution.

3.9.2. *Update HS harmony memory.* Based on the objective functions, HS algorithm evaluates the new harmony solution. If the fitness value of the solution is better than the worst fitness value in HM, it will include that new solution in HM and exclude the worst solution from HM.

3.10. **Termination criteria.** This research uses the number of generations as the termination condition. The number of generation set for this research is 75 which is based on the size of the datasets as well as the generation number used by the previous researches on the same datasets.

4. **Experimental Results and Discussion.**

To illustrate the proposed algorithm the dataset 8 generated by Mukhopadhyay et al. [27] is adopted and the data is given in Table 1. We are using this specific example to ease comparative understanding of the H-CCGaHs against existing algorithms as this example has already been applied by previous researchers. From the table, there are seven part types to be loaded on four machines.

All of the machines are assumed to have one shift, i.e., 480 minutes available time and all of them have five tool slots each.

The time taken to find solutions is not only dependent on the size of population but also on the tightness of the constraints and the approach used in designing the chromosome representation as well as the effectiveness of the objective functions. This is because substantial processing time is required to filter non-feasible solutions. The work is hoped to manage the machine utilization as well as allowing more part types to be scheduled using the same resources. The chromosome representation ensures that the values are representing the right combinations which only allow the valid machines to be allocated to the respective part types. In addition, the objective function will also check on the chromosome to ensure the right combination as well the total amount of time allocated does not exceed the remaining time available for the machine.

The various combination of parameter values are used to conduct a sensitivity analysis on the sample data set to identify the optimal control parameters for the proposed H-CCGaHs (Table 2). Exhaustive computations have been carried out to assess the effectiveness of the proposed algorithm and its performance is compared with the previous studies. From the experiments, options 2 and 6 are giving the overall best result where higher HMCR and PAR rate will improve the solution. Higher HMCR and PAR rates promote higher probability of selecting the new sequence based on historical position of solution stored in HM as well as improvise the solution and avoid being trapped in local optima. In addition, MR rate is not effectively affecting the result.

The proposed approach has been implemented using C# compiler. It is tested on the benchmark problems available in Mukhopadhyay et al. [27]. The solution for the 10 datasets and comparative results between the proposed algorithm and existing literatures are given in Table 3. The comparative study is performed between proposed H-CCGaHs with heuristics developed by [3, 8, 15, 28, 29, 30, 31, 32]. From the result it can be observed that the proposed algorithm H-CCGaHs performs better than most of the heuristics available in the literature for the minimization of system unbalance as well as combined objective functions (COF). The average COF (1.8160) in the last row of Table 3 shows that H-CCGaHs is superior in performance over the other heuristics considered for the evaluation. It is significant to note that dataset 4 already achieved optimum throughput for all the literatures compared, whereas dataset 9 met its optimum throughput through our previous works ([31, 32]) and proposed algorithm. However, only the proposed H-CCGaHs algorithm is able to supersede other algorithms in achieving optimum throughput for dataset 2, which the throughput produced is the same with the batch size.

TABLE 2. Control parameters of the proposed algorithm and overall result

Control Parameter	Option					
	1	2	3	4	5	6
Population/HM	20	20	20	20	20	20
No of Generation	75	75	75	75	75	75
Crossover	0.70	0.70	0.70	0.70	0.70	0.70
Mutation	0.01	0.10	0.30	0.01	0.10	0.30
HMCR	0.85	0.95	0.90	0.90	0.85	0.95
Pitch Adjustment	0.10	0.10	0.01	0.01	0.10	0.10
Overall fitness	1.8098	1.816	1.8158	1.8158	1.8158	1.816

TABLE 3. Comparison of proposed H-CCGaHs with other heuristics

		[28]		[8]		[33]		[29]		[30]		[15]		[31]		[32]		Proposed H-CCGaHs	
Data	Batch	TH	SU	TH	SU	TH	SU	TH	SU	TH	SU	TH	SU	TH	SU	TH	SU	TH	SU
Set	Size	COF		COF		COF		COF		COF		COF		COF		COF		COF	
1	80	42	76	48	14	44	127	48	14	48	14	48	14	52	0	52	0	52	0
		1.4854		1.5927		1.4839		1.4615		1.5927		1.5927		1.6500		1.6500		1.6500	
2	73	63	234	46	18	63	124	46	18	63	124	63	22	64	15	64	15	73	30
		1.7411		1.6208		1.7984		1.6208		1.7984		1.8516		1.8689		1.8689		**1.9844**	
3	79	69	152	69	72	73	128	69	94	69	72	73	28	73	28	73	28	73	28
		1.7943		1.8359		1.8574		1.8245		1.8359		1.9095		1.9095		1.9095		1.9095	
4	51	51	819	51	819	51	819	51	819	51	819	51	819	51	819	51	819	51	819
		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734	
5	76	61	264	53	187	61	264	53	175	61	264	61	264	61	69	61	69	61	69
		1.6651		1.6		1.6651		1.6062		1.6651		1.6651		1.7667		1.7667		1.7667	
6	73	63	214	61	28	63	284	64	69	61	28	64	37	64	7	64	7	64	7
		1.7516		1.821		1.7151		1.8408		1.821		1.8731		1.8731		1.8731		1.8731	
7	78	48	996	54	165	54	177	54	165	63	231	63	231	63	21	63	21	63	21
		1.0966		1.6064		1.6001		1.6064		1.6874		1.6874		1.7968		1.7968		1.7968	
8	70	43	158	48	63	44	13	44	13	44	13	48	63	54	56	54	56	54	56
		1.532		1.6529		1.6218		1.6218		1.6218		1.6529		1.7423		1.7423		1.7423	
9	88	88	309	88	309	88	309	88	309	88	309	88	309	88	56	88	56	88	56
		1.8391		1.8391		1.8391		1.8391		1.8391		1.8391		1.9708		1.9708		1.9708	
10	67	55	166	56	122	56	122	54	82	56	122	56	122	67	205	67	205	67	205
		1.7344		1.7723		1.7723		1.7633		1.7723		1.7723		1.8932		1.8932		1.8932	
COF		1.6213		1.6914		1.6927		1.6889		1.7207		1.7417		**1.8045**		**1.8045**		**1.8160**	

For all of the machine loading datasets, we have provided the detailed description regarding the obtained results, and operation-machine allocation combinations in Table 4. The table shows the COF result, throughput and system unbalance for each data set and the respective machine assignment for part type for each operation. The '–' indicates that the respective operation is not required by the part type, while 'na' means that the particular part type is not being assigned due to resource constraint.

Sarma et al. [34] recommended using comparative improvement index (CII) for performance comparison. Therefore, CII is adopted here to investigate the % improvement in the results obtained by the proposed algorithm. We do not compare the comparative improvement index of system unbalance (CII_{SU}) and the comparative improvement index of throughput (CII_{TH}) individually since the increase of CII_{TH} may decrease CII_{SU} and vice versa. So it is more appropriate to use CII_{COF} and CII_{THMax} to indicate the comparative improvement. CII_{COF} and CII_{THMax} are obtained by using the following:

$$CII_{COF} = \frac{(COF_{Hy} - COF_H)}{COF_{Hy}} * 100 \tag{13}$$

$$CII_{THMax} = 100 - \left(\frac{(TH_T - TH_{Hy})}{TH_T} * 100 \right) \tag{14}$$

where

CII_{COF}	the comparative improvement in the combined objective functions
CII_{THMax}	the percentage achievement based on the maximum achievable throughput
COF_H	the combined objective function obtained by heuristic procedures [3, 8, 15, 28, 29, 30, 31, 32]

COF_{Hy}	the combined objective function obtained by the proposed algorithm
TH_T	maximum throughput in each dataset
TH_{Hy}	the throughput obtained by the proposed algorithm

The corresponding CIIs values are given in Table 5. For the sake of comparison, CII_{COF} and CII_{THMax} are compared with heuristic procedure performed by [3, 8, 15, 28, 29, 30, 31, 32]. From the table, it can been observed that for almost all the problems, the proposed algorithm shows a considerable improvement. In the table, the term given by the '0' sign in CII_{COF} with 100% in CII_{THMax} indicates for that problem optimality (i.e. maximum possible throughput and minimum possible system unbalance) has been achieved. The '0' sign in CII_{COF} with non-100% in CII_{THMax} indicates no improvement. Note that for dataset 2, the proposed H-CCGaHs manages to achieve 100% maximum throughput as compared with the other algorithms. For the description of all 10 datasets of the underlying problem, readers are requested to refer to [27, 28].

Figure 6 shows the comparison with other literature in terms of achieving 100% total throughput. From the graph, it is shown that datasets 4 and 9 have already reached 100% total throughput from the beginning, while other datasets still have room for improvement. For dataset 10, the algorithms developed by our previous and current studies (CCGA, HS and H-CCGaHs) achieve 100% total throughput. However, only the proposed algorithm (H-CCGaHs) is able to achieve 100% total throughput for dataset 2.

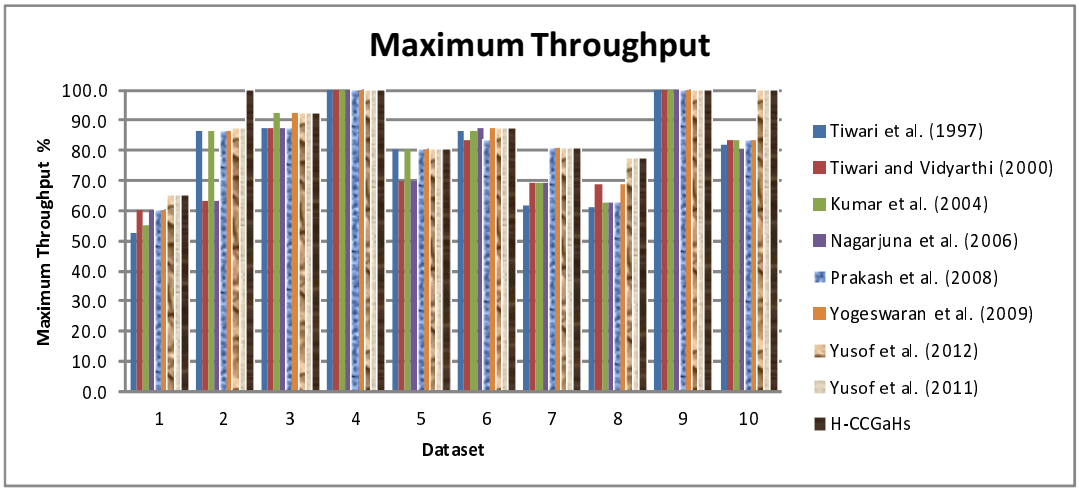


FIGURE 6. Maximum throughput comparisons

Table 6 shows the convergence (number of generations) and duration (in seconds) taken to run each data set for the proposed three algorithms. From the table, overall H-CCGaHs is better in achieving convergence, while HS is the worst. In term of duration, H-CCGaHs is relatively better than CCGA, while HS run the fastest. Thus, it is clear that H-CCGaHs is the best method to achieve a fast convergence with the moderate time taken to run the algorithm, with the consideration of the overall good results obtained.

5. **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.

TABLE 4. Operation-machine allocation combination for the assigned part sequence

Data set	Part Type	operation/machine 1	2	3
1	COF: 1.6500	TH: 52	SU: 0	
	1	3	–	–
	6	4	3	2
	3	1	3	–
	5	2	2	–
	7	2	1	4
	2	na	na	–
	8	na	na	na
	4	na	na	–
2	COF: 1.9843	TH: 73	SU: 30	
	1	4	2	–
	3	1	–	–
	2	1	4	–
	4	3	–	–
	5	4	1	–
	6	2	–	–
3	COF: 1.9095	TH: 73	SU: 28	
	1	2	–	–
	2	2	–	–
	3	4	1	–
	5	4	3	–
	4	na	–	–
4	COF: 1.5734	TH: 51	SU: 819	
	5	1	–	–
	4	3	3	2
	1	2	1	4
	3	4	–	–
	2	4	3	–
5	COF: 1.7667	TH: 61	SU: 69	
	6	1	–	–
	1	2	2	–
	2	1	–	–
	4	4	2	3
	3	na	na	–
	5	3	–	–

Data set	Part Type	operation/machine 1	2	3
6	COF: 1.8731	TH: 64	SU: 7	
	1	2	–	–
	3	1	–	–
	2	1	3	–
	5	3	1	–
	6	2	–	–
	4	na	–	–
7	COF: 1.7968	TH: 63	SU: 21	
	6	2	2	
	2	4	2	3
	1	1	–	–
	3	1	–	–
	5	3	–	–
	4	na	na	
8	COF: 1.7423	TH: 54	SU: 56	
	5	4	3	–
	1	2	3	–
	7	4	2	–
	2	3	–	–
	4	na	–	–
	6	3	2	1
	3	na	na	na
9	COF: 1.9708	TH: 88	SU: 56	
	2	4	–	–
	4	3	4	–
	7	1	4	–
	6	4	1	–
	1	3	1	–
	3	2	–	–
	5	2	3	–
10	COF: 1.8932	TH: 67	SU: 205	
	6	3	1	–
	2	3	4	3
	4	3	4	1
	3	1	2	–
	5	1	2	4
	1	3	4	2

The main contribution of the present research is an efficient heuristic approach based on hybrid of CCGA and HS algorithm (H-CCGaHs), which is able to solve machine-loading problem of FMS. While some of the previous studies considered part type sequence and operation allocation as separate but interconnected components, the present research

TABLE 5. CII_{COF} and CII_{THMax} comparisons with other heuristics

Dataset	Batch Size		[28]	[8]	[3]	[29]	[30]	[15]	[31]	[32]	Proposed H-CCGaHs
1	80	CII_{THMax}	52.5	60.0	55.0	60.0	60.0	60.0	65.0	65.0	65.0
		CII_{COF}	11.1	3.6	11.2	3.6	3.6	3.6	0	0	
2	73	CII_{THMax}	86.3	63.0	86.3	63.0	86.3	86.3	87.7	87.7	100
		CII_{COF}	14.0	22.4	10.3	22.4	10.3	7.2	6.2	6.2	
3	79	CII_{THMax}	87.3	87.3	92.4	87.3	87.3	92.4	92.4	92.4	92.4
		CII_{COF}	6.4	4.0	2.8	4.7	4.0	0	0	0	
4	51	CII_{THMax}	100	100	100	100	100	100	100	100	100
		CII_{COF}	0	0	0	0	0	0	0	0	
5	76	CII_{THMax}	80.3	69.7	80.3	69.7	80.3	80.3	80.3	80.3	80.3
		CII_{COF}	6.1	10.4	6.1	10.0	6.1	6.1	0	0	
6	73	CII_{THMax}	86.3	83.6	86.3	87.7	83.6	87.7	87.7	87.7	87.7
		CII_{COF}	6.9	2.9	9.2	1.8	2.9	0	0	0	
7	78	CII_{THMax}	61.5	69.2	69.2	69.2	80.8	80.8	80.8	80.8	80.8
		CII_{COF}	63.9	11.9	12.3	11.9	6.5	6.5	0.0	0.0	
8	70	CII_{THMax}	61.4	68.6	62.9	62.9	62.9	68.6	77.1	77.1	77.1
		CII_{COF}	13.7	5.4	7.4	7.4	7.4	5.4	0	0	
9	88	CII_{THMax}	100	100	100	100	100	100	100	100	100
		CII_{COF}	7.2	7.2	7.2	7.2	7.2	7.2	0.0	0.0	
10	67	CII_{THMax}	82.1	83.6	83.6	80.6	83.6	83.6	100.0	100.0	100.0
		CII_{COF}	9.2	6.8	6.8	7.4	6.8	6.8	0	0	
Average COF			12.0	7.4	7.3	7.5	5.5	4.3	0.6	0.6	

TABLE 6. Convergence and duration taken to run comparison

	Convergence (number of generations)			Duration (in second)		
Dataset	CCGA	HS	H-CCGaHs	CCGA	HS	H-CCGaHs
1	9	21	26	11.53	9.52	7.66
2	9	24	6	7.42	3.25	5.22
3	7	4	8	5.75	2.47	4.43
4	6	5	7	7.23	2.33	4.36
5	7	6	8	9.33	3.73	6.23
6	9	18	12	4.36	2.86	4.44
7	7	9	12	6.28	3.33	5.71
8	8	35	6	7.39	5.56	6.64
9	6	7	6	6.84	5.22	6.81
10	8	17	9	6.65	3.59	6.46

adopts an approach of treating part type sequencing and machine allocation problem as one main goal. Because of this, 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 is being identified.

Exhaustive computations are carried out to assess the effectiveness of the proposed algorithm, and its performance is compared with the previous studies. From the results,

the proposed H-CCGaHs offers better results for most of the test problems, in which the COF increased by 4.27% compared with the best result of the other heuristic methods [15]. 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 the early number of generations. This ability is very important for the manufacturing industry that always prioritize to use various methods that can save processing time and cost in generating machine allocation planning. The results also show that most of the superior results occur on the datasets 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 resources availability and constraints. Although the proposed algorithm is tested on the datasets available in the open literature, this work can be extended to test on large scale real life problems. It can also be extended to solve multi-objective machine loading problems and is able to handle 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/6313026. 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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] S. K. Mukhopadhyay, B. Maiti and S. Garg, Heuristic solution to the scheduling problems in flexible manufacturing system, *International Journal of Production Research*, vol.29, no.10, pp.2003-2024, 1991.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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, pp.4753-4764, 2009.

[15] 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.

[16] Z. W. Geem, J. H. Kim and G. Loganathan, A new heuristic optimization algorithm: Harmony search, *SIMULATION*, vol.76, no.2, pp.60-68, 2001.

[17] Z. W. Geem, School bus routing using harmony search, *Genetic and Evolutionary Computation Conference*, 2005.

[18] Z. Geem, Improved harmony search from ensemble of music players, in *Lecture Notes in Computer Science*, B. Gabrys, R. Howlett and L. Jain (eds.), Springer Berlin/Heidelberg, 2006.

[19] M. Al-Betar and A. Khader, A harmony search algorithm for university course timetabling, *Annals of Operations Research*, pp.1-29, 2010.

[20] A. Mukhopadhyay, A. Roy, S. Das and A. Abraham, Population-variance and explorative power of harmony search: An analysis, *The 3rd International Conference on Digital Information Management*, pp.775-781, 2008.

[21] O. Alia, R. Mandava and M. Aziz, A hybrid harmony search algorithm to mri brain segmentation, *The 9th IEEE International Conference on Cognitive Informatics*, pp.712-721, 2010.

[22] L. Wang, Q.-K. Pan and M. F. Tasgetiren, A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem, *Computers & Industrial Engineering*, vol.61, no.1, pp.76-83, 2011.

[23] L. Li and L. Wang, Hybrid algorithms based on harmony search and differential evolution for global optimization, *Proc. of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, New York, NY, USA, pp.271-278, 2009.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] A. Prakash, N. Khilwani, M. Tiwari and Y. Cohen, Modified immune algorithm for job selection and operation allocation problem in flexible manufacturing systems, *Advances in Engineering Software*, vol.39, pp.219-232, 2008.

[31] U. K. Yusof, R. Budiarto and S. Deris, Constraint-chromosome genetic algorithm for flexible manufacturing system machine-loading problem, *International Journal of Innovative Computing, Information and Control*, vol.8, no.3(A), pp.1591-1609, 2012.

[32] U. K. Yusof, R. Budiarto and S. Deris, Harmony search algorithm for flexible manufacturing system (fms) machine loading problem, *The 3rd Conference on Data Mining and Optimization (DMO)*, Kuala Lumpur, Malaysia, pp.26-31, 2011.

[33] R. Kumar, A. K. Singh and M. K. Tiwari, A fuzzy based algorithm to solve the machine-loading problems of a-fms and its neurofuzzy petri net model, *International Journal of Advanced Manufacturing Technology*, vol.23, pp.318-341, 2004.

[34] U. M. B. S. Sarma, S. Kant, R. Rai and M. 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.