Applied Artificial Intelligence

An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

Scheduling of Virtual Cellular Manufacturing Systems: A Biogeography-Based Optimization Algorithm

M. Zandieh

To cite this article: M. Zandieh (2019) Scheduling of Virtual Cellular Manufacturing Systems: A Biogeography-Based Optimization Algorithm, Applied Artificial Intelligence, 33:7, 594-620, DOI: 10.1080/08839514.2019.1577021

To link to this article: https://doi.org/10.1080/08839514.2019.1577021

Published online: 20 Mar 2019.

APPLIED ARTIFICIAL INTELLIGENCE
2019, VOL. 33, NO. 7, 594–620
https://doi.org/10.1080/08839514.2019.1577021

Scheduling of Virtual Cellular Manufacturing Systems: A Biogeography-Based Optimization Algorithm

M. Zandieh

Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G.C., Tehran, Iran

ABSTRACT
Virtual cellular manufacturing system (VCMS) is one of the modern strategies in the production facilities layout, which has attracted considerable attention in recent years. In this system, machines are located in different positions on the shop floor and virtual cells are a logical grouping of machines, jobs, and workers from the viewpoint of the production control system. These features not only enhance the system's agility but also allow a dynamic reassignment of cells as demand changes. This paper addresses the VCMS scheduling problems where the jobs have different orders on machines and the objective is to *simultaneously* minimize the weighted sum of the makespan and total traveling distance in order to create a balance between criteria. The research methodology firstly consists of a mathematical programming model with regard to the production constraints in order to describe the characteristics of the VCMS. Secondly, a basic genetic algorithm (GA), a biogeography-based optimization (BBO) algorithm, an algorithm based on hybridization of BBO and GA, and the BBO algorithm accompanied by restart phase are developed to solve the VCMS scheduling problems. The developed algorithms have been compared to each other and their performance are evaluated in terms of their best solution and computational time as effectiveness and efficiency criteria, respectively. Consequently, the performance of the best algorithm has been evaluated by the state-of-the-art algorithm, GA, in the literature. The results show that the best algorithm based on BBO could find solutions at least as good as the last famous algorithm, GA, in the literature.

Introduction

In the competitive industrial environment, it is imperative that production systems be controlled optimally. This is enforcing manufacturing firms to reduce production costs and to meet the production schedules. This presents the need for costly and time-consuming reorganization of the shop floor and allows for quick response to changing environment and demands (Nomden, Slomp, and Suresh 2006).

CONTACT M. Zandieh m_zandieh@sbu.ac.ir Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G.C., Tehran, Iran
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/uaai.

The latest facility layouts such as product layout and process layout sequentially limit the flexibility and efficiency of the production systems. So in the 1960s, cellular manufacturing system (CMS) based on group technology (GT) principles, was introduced to cope with the drawbacks of product and process layouts. The advantages of CMS are emphasized in the literature including the reduction in cycle time, number of setups, work-in-process, and delivery time along with increasing flexibility, production control, and improving product quality, which results in the reduction in production costs (Mansouri, Moattar Husseini, and Newman 2000). Although the CMS combines the efficiency of flow shop environment with the flexibility of job shop manufacturing, the efficiency of cellular layout tends to decline drastically as soon as changes occur in the pattern of products' demand (Wemmerlow and Hyer 1989). A relative new alternative has been considered in recent years, namely virtual cellular manufacturing system (VCMS).

The virtual manufacturing cell (VMC) is specified as data files and processors in a controller, not as a fixed physical grouping of machines. It means that when a job order needs a group of machines, a virtual cell controller takes over the control of these machines and makes feasible interrelations between them. The controller will then control the machines in the newly formed cell until the job is completed. At the completion of the job, the VMC is terminated, and the machines will be released for other production orders. The logical grouping of machines, jobs, and workers is based on predefined logic, and it is only resident in the production control system and in the minds of the workers (Slomp, Chowdary, and Suresh 2005). In the VCMS, each job has an individual processing order with more than one route, so that the problem resembles a flexible job-shop scheduling problem (FJSP). Although in the FJSP, it is assumed that individual machines are working in parallel, the distance between machines is ignored (Kessen, Sanchoy, and Gungor 2010).

In this paper, a mathematical programming model is developed to describe the characteristics of a VCMS, while several algorithms based on biogeography concepts are developed to solve the production scheduling of jobs in the VCMS. The biogeography-based optimization (BBO) algorithm is a new evolutionary algorithm for global optimization that was firstly introduced by Simon (2008). Mathematical modeling of biogeography concept describes how species migrate from one habitat to another habitat, how new species arise, and how species become extinct. In addition, the new BBO algorithm, an algorithm based on hybridization of BBO and genetic algorithm (GA), and the BBO algorithm along with a restart phase are developed to find good quality solutions in a shorter time, compared to the last famous heuristic algorithm, GA, of the literature, which is considered as the research gap.

The rest of the paper is organized as follows: Section 2 reviews the literature mostly dealing with the problem of interest, i.e., the VCMS. The problem is well described in section 3 with the help of the detailed mathematical programming model. Section 4 introduces the fundamental of the BBO algorithm and its implementation to solve production scheduling problems. The proposed search algorithms are comprehensively described in this section. Section 5 represents the experimental results obtained from both the GA and three algorithms based on BBO and the comparison of the performance of these algorithms. Apart from this, a comparison between the best BBO algorithm and the state-of-the-art algorithm in the literature is represented in this section. Finally, in section 6, a concise conclusion about the research along with some suggestions for future works are presented.

Literature Review

The virtual manufacturing concept was first developed at National Bureau of Standards to address specific control problems encountered in the design phase of automated manufacturing of small batches of machined parts (McLean, Bloom, and Hopp 1982). Rheault, Drolet, and Abdulnour (1995) introduced the concept of a dynamic cellular manufacturing system (DCMS) for solving the cell reconfiguration problem. They developed an integer programming model to determine the location of workstations with the objective of minimizing the total material handling costs. The DCMS has been investigated by many researchers with respect to dynamic cell reconfiguration, even over a multi-period planning horizon (Ahkioon, Bulgak, and Bektas 2009; Balakrishnan and Cheng 2005; Defersha and Chen 2006; Kia et al. 2012; Mahdavi et al. 2010).

Vakharia, Moily, and Huang (1999) compared the performance of virtual cells and multi-stage flow shops through analytical approximations. They stated that virtual cells need for the dedication of individual machines within the current departments to a specific sort of part families. The results indicate that the ratio of setup time to run time per batch is a major factor that influences the decision to implement virtual cells. Saad, Baykasoglu, and Gindy (2002) explored the feasibility of virtual cells as a strategy to solve the cell configuration problem. They were developed a tabu search algorithm to reconfigure the manufacturing cells without physical relocation and a simulation package was used to appraise the performance of the integrated manufacturing system.

Ko and Egbelu (2003) proposed an algorithm to solve the machine cell creation problem by determining optimal component routing. They stated the number of the VMC was depended on the components' attributes. Also, they demonstrated the concept of sharing workstations in manufacturing cells among multiple part families. Ko and Egbelu (2000) proposed a methodology for designing the VMC. They compared dynamic and static manufacturing systems at the intent of examining the influence of variations in the product mix on the shop performance. The performance measures were considered as total setup time and total material handling distances. They illustrate the superiority of a dynamic CMS to a static CMS.

Kannan and Ghosh (1996a, 1996b) and Kannan (1998) studied many part family-based scheduling rules in process layout. They represented the perfection of VMCs due to the ability to combine setup time efficiency of cellular layout and routing flexibility of process layout. They also stated that a machine can be altered without significant modification on the shop floor, and consequently, VMCs are able to respond to demand changes quickly. Drolet, Marcoux, and Abdulnour (2008) developed a linear programming model for scheduling VMCs with the objective function of minimizing total traveling distance. In this study, the planning horizon is divided into periods and the solution time of the model is susceptible to the number of periods, because successive operations must start at the beginning of periods. If the number of periods is selected too large, unnecessary waiting times appear. On the other hand, if selected too small, solution time hikes rapidly.

Mak et al. (2009) developed a non-linear programming formulation for scheduling VMCs with respect to time period structure. Because of the intractability of the non-linear model, they suggested a solution by using ant colony optimization (ACO). Kessen et al. (2009) developed an ant colony optimization-based meta-model to reflect the behaviors of virtual cells, process layout, and cellular layout. They addressed the virtual cells by using family-based scheduling rule and compared these three systems by simulation package. They also developed a multi-objective mixed integer programming formulation to characterize the scheduling of the VCMS problem. The objective function was the weighted sum of makespan and total traveling distance. Since the nature of the problem was too complex, they have developed a GA approach finding good quality solutions.

Recently, Baykasoglu and Gorkemli (2017) addressed a new VCMS with dynamic demand arrivals with the help of agent-based modeling approach. The proposed approach is able to form a part family, cell formation, and scheduling phase, *simultaneously*. Aalaei and Davoudpour (Aalaei 2016) proposed a bi-objective model for a dynamic VCMS and supply chain design with respect to important manufacturing features such as multi-market allocations, multi-period production planning, multi-plants, and facility locations under uncertainty conditions. Hamedi and Esmaeilian (2015) studied the effectiveness of functional and distributed layouts on capability-based VCMS performance with the help of a multi-objective mathematical model and a multi-objective tabu search. Yang et al. (2016) proposed an exploratory study of a virtual cell design for thin-film-transistor-liquid crystal display (TFT-LCD) array manufacturing. The virtual cell design is dynamically reconfigured based on the product mix changes to retain the efficiency and effectiveness of the system. Tambuskar, Narkhede, and Mahapatra (2015) proposed a novel algorithm using discrete particle swarm optimization technique to design virtual cells by considering processing time, operation sequence, routing flexibility, machine capacity, machine flexibility, and demand. Fu and Murata (2016) studied the configuration design of the VCMS with batch splitting operations. They proposed an effective methodology to form novel virtual cells and operation take care of allocation of shared machines and batch size of parts.

Problem Description and Formulation

We consider the problem of scheduling n jobs $N = \{1, 2, \ldots, n\}$ with different processing orders on m machines $M = \{1, 2, \ldots, m\}$ including $\mathcal{L}$ types $I = \{1, 2, \ldots, \mathcal{L}\}$ $(\mathcal{L} \leq m)$ in the VCMS. Each machine type $i \in I$ has one or more individual machine $V = \{1, 2, \ldots, v_i\}$ $(v_i < m,\ i \in I)$. s_{iv} $(i \in I$ and $v \in V)$ indicates v^{th} machine of machine type $i \in I$ and v_i is the total number of machine type $i \in I$ $(\sum_{i \in I} v_i = m)$. All the machines belonging to the machine type $i \in I$ are located at different positions on the shop floor and have the same machine capabilities. Figure 1 shows the schematic representation of the VCMS.

Each job $j \in N$ has one or more fixed operation $H = \{1, 2, \ldots, h_j\}$, where $O_{j,h}$ represents the pre-determined operation $h \in H$ of the job $j \in N$. Each operation $O_{j,h}$ has a processing time $P_{j,h,i}$ on machine type i $(j \in N$, $i \in I$, and $h \in H)$. Jobs are produced as batches without batch splitting and $N_j$represent the batch size of the job $j \in N$. A transportation cost $D_{s_{iv}, s_{i'v'}}$ $(i, i' \in I, i \neq i'$ and $v, v' \in V)$ is incurred based on traveling distance between machines

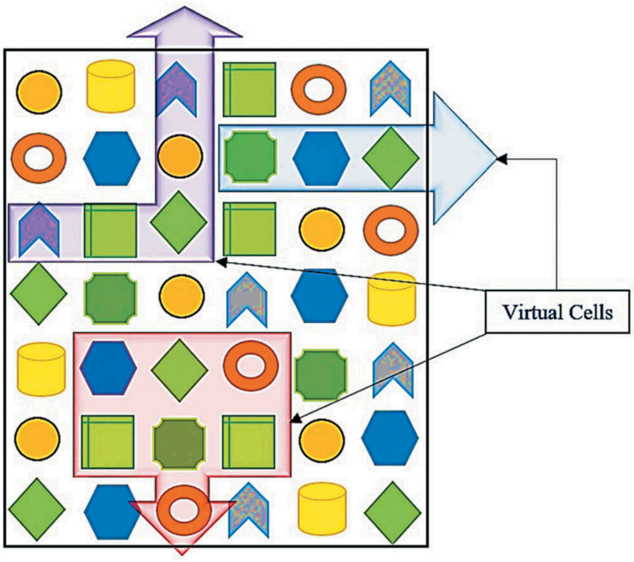


Figure 1. Schematic representation of the VCMS concept and design.

whenever each job $j \in N$ is carried from machine s_{iv} to machine $s_{i'v'}$. Processing times, batch sizes, and traveling distance between machines are pre-determined. All jobs are available at the beginning of the planning horizon. Since each machine $k \in M$ processes a job at a time, the jobs assigned to a machine are processed in an order. Thus, each machine $k \in M$ can process at most ℓ_{iv} jobs $L = \{1, 2, \ldots, \ell_{iv}\}$ so that each job $j \in N$ is processed in the order l ($l \in L$). ℓ_{iv} indicates the maximum number of jobs which can be processed by v^{th} machine of machine type $i \in I$. All machines are immobile and breakdown, as well as maintenance activities are ignored. Preemption in the model is not allowed.

Programming Formulation

Define the following binary variables for $j \in N$, $h \in H$, $i \in I$, $v \in V$ and $l \in L$. The variable $Y_{s_{iv},j,h} = 1$ if the machine s_{iv} is assigned for operation $o_{j,h}$ of the job j; $Y_{s_{iv},j,h} = 0$, otherwise. The variable $X_{s_{iv},j,h,l} = 1$ if the operation $o_{j,h}$ is performed on the machine s_{iv} in the order l; $X_{s_{iv},j,h,l} = 0$, otherwise. C_{max} is makespan which is equal to the completion time of the last job processed on the current planning horizon. $T_{s_{iv},l}$ is the starting time of any job processed in the order l on machine s_{iv}, while $t_{j,h}$ is the starting time of the processing h^{th} operation of the job j. In the following programming model, two main decisions including machine assignment and operation sequencing on machines are integrated for solving a job scheduling problem in the VCMS.

The objective function is a linear combination of makespan and the total traveling distance of jobs with the aim of creating a balance between two criteria. Both objectives favor the producer's interest by minimizing the production cost including work-in-process (WIP) inventory, inventory holding costs, energy consumption, and jobs' movements. Based on manufacturing companies' policy, two criteria might have different importance. Particularly, the second criterion considers the operation assignment of the jobs to machines in terms of the traveling distance of the jobs between machines. Therefore, these two criteria are combined with the help of normalized importance/weighted coefficients; α and β are weights attributed to the first and second criterion of the objective function, respectively. As they are assumed to be normalized, α and β should add to 1.

$$\min(\alpha C_{max} + \beta \sum_{jN} \sum_{h} \sum_{iI} \sum_{\substack{i' \epsilon I \\ i' \neq i}} \sum_{v \in V} \sum_{v' \in V} Y_{s_{iv},j,h} Y_{s_{i'v'},j,h+1} D_{s_{iv},s_{i'v'}} N_j)$$

and the operational constraints of the research problem are explained as follows:

$$C_{max} \geq t_{j,h} + \sum_{i \in I} \sum_{v \in V} Y_{s_{iv},j,h} P_{j,h,i} N_j, \quad \forall j \in N, h \in H \tag{1}$$

Constraint set (1) relates to the makespan value of the system. This constraint guarantees that C_{max} must be equal or greater than the completion time of all operations of all jobs.

$$t_{j,h} + \sum_{i \in I} \sum_{v \in V} Y_{s_{iv},j,h} P_{j,h,i} N_j \leq t_{j,h+1}, \quad \forall j \in N, h \in H \tag{2}$$

Constraint set (2) is incorporated into the precedent relationship of jobs. This constraint ensures that the operation of each job in a machine cannot be started until it has been completely processed on a prior machine, where the job had its latest operation.

$$T_{s_{iv},l} + X_{s_{iv},j,h,l} P_{j,h,i} N_j \leq T_{s_{iv},l+1}, \quad \forall j \in N, h \in H, i \in I, v \in V, l \in L \tag{3}$$

Constraint set (3) is incorporated in the processing order of jobs on a machine. This constraint ensures that successive operations of any machine must wait until the preceding one has been completely processed.

$$t_{j,h} + M\left(1 - X_{s_{iv},j,h,l}\right) \geq T_{s_{iv},l}, \quad \forall j \in N, h \in H, i \in I, v \in V, l \in L \tag{4}$$

$$T_{s_{iv},l} + M\left(1 - X_{s_{iv},j,h,l}\right) \geq t_{j,h}, \quad \forall j \in N, h \in H, i \in I, v \in V, l \in L \tag{5}$$

Constraint (4) together with constraint (5) guarantee that if $X_{s_{iv},j,h,l} = 1$, then the starting time of $O_{j,h}$ and l^{th} order on the machine s_{iv} must be the same. If $X_{s_{iv},j,h,l} = 0$, then the large constant M renders the constraint redundant.

$$\sum_{j \in N} \sum_{h \in H} X_{s_{iv},j,h,l} \leq 1, \quad \forall i \in I, v \in V, l \in L \tag{6}$$

Constraint set (6) guarantees that each order of any machine is assigned to at most one operation of all jobs.

$$\sum_{i \in I} \sum_{v \in V} Y_{s_{iv},j,h} = 1, \quad \forall j \in N, h \in H \tag{7}$$

Constraint set (7) restricts that each operation of each job is assigned to exactly one machine among competitive ones.

$$\sum_{l \in L} X_{s_{iv},j,h,l} = Y_{s_{iv},j,h}, \quad \forall j \in N, h \in H, i \in I, v \in V \tag{8}$$

Constraint set (8), known as a balance constraint, is incorporated into the model to establish a balance between machine assignment and order assignment of jobs on machines. In other words, an operation of a job is processed by a machine only when this operation is positioned on any order $l \in L$ of the machine.

$$C_{max} \geq 0, \quad t_{j,h} \geq 0, \; T_{s_{iv},l} \geq 0, \quad X_{s_{iv},j,h,l} \in \{0,1\}, \quad Y_{s_{iv},j,h} \in \{0,1\} \tag{9}$$

Finally, constraint set (9) defines integer and binary variables. Although the second part of the objective function makes the model non-linear, with the help of the following new variable shown by Equation (10) as well as some extra constraint shown by Equations (11) and (12), the non-linearity of the model can be eliminated.

$$Z_{s_{iv},s_{i'v'},j,h,h+1} = Y_{s_{iv},j,h}.Y_{s_{i'v'},j,h+1} \tag{10}$$

$$Z_{s_{iv},s_{i'v'},j,h,h+1} - Y_{s_{iv},j,h} - Y_{s_{i'v'},j,h+1} + 1 \geq 0, \; \left(o_{j,h}, \; s_{iv}\right) \to \left(o_{j,h+1}, \; s_{i'v'}\right) \tag{11}$$

$$2.\, Z_{s_{iv},s_{i'v'},j,h,h+1} - Y_{s_{iv},j,h} - Y_{s_{i'v'},j,h+1} \leq 0, \; \left(o_{j,h}, \; s_{iv}\right) \to \left(o_{j,h+1}, \; s_{i'v'}\right) \tag{12}$$

Meta-Heuristic Algorithm in the VCMS Problem

Since the problem is NP-hard, heuristic optimization is an approach to solving complex problems. A basic population-based meta-heuristic such as GA or Artificial Bee Colony (ABC) (Garg 2014), a basic local search meta-heuristic such as Tabu Search (TS) (Shahvari and Logendran 2015; Shahvari, Salmasi, and Logendran 2009; Shahvari et al. 2012), a robust version of basic meta-heuristic algorithms such as stage-based TS (Shahvari and Logendran 2016a, 2017), and a hybridization of two local search structures, two population-based structures such as Particle Swarm Optimization and GA (PSO/GA) (Garg 2016a) or local search and population-based structures such as Tabu Search/Path-Relinking (TS/PR), Genetic Algorithm/Gravitational Search Algorithm (GA/GSA), and PSO/TS (Garg 2015a; Shahvari and Logendran 2016b) can be implemented to find the optimal/near optimal solution for the research problem.

Based on the characteristics of the research problem along with preliminary experiments, a basic local search algorithm cannot present good quality solutions compared to a basic population-based algorithm. Apart from this, a hybridization of a local search and population-based structures is an inefficient algorithm compared to a hybridization of two population-based structures. As a result, a population-based algorithm or a hybridization of two population-based algorithms can present better results. This being the case, we decided to develop an appropriate basic population-based algorithm in terms of the characteristics of the search problem, i.e., BBO along with the other derivatives of the BBO algorithm such as a robust BBO, i.e., BBO accompanied by a restart phase (BBO/RF), and a hybrid of two population-based algorithms, i.e., BBO/GA. Then, in order to show the superiority of developed BBO algorithms, they are compared with another well-known population-based algorithm, GA, and the results of all developed algorithms are evaluated with the help of the best solutions obtained by all algorithms.

BBO is an evolutionary algorithm for global optimization that is based on the science of biogeography. Biogeography is the study of the distribution of species (animals and plants) over time and space (Du 2009). The environment of BBO corresponds to an archipelago, where every possible solution for the optimization problem is an island. The island is a habitat that is geographically separated from other habitats. In this paper, island and habitat are considered the same. BBO is modeled after the immigration and emigration of species between these habitats. The application of this idea for optimization problems allows information sharing between candidate solutions (Garg 2015b, 2016b; Rajasomashekar and Aravindhababu 2012; Simon, Shah, and Scheidegger 2013).

BBO Algorithm in General

In each generation of BBO, the initial population generated randomly is not discarded. As another distinction, for each generation, BBO uses the fitness of each solution to determine its migration (immigration and emigration) rates (Ma and Simon 2011). Each solution feature is called a suitability index variable (SIV), while the goodness of each solution is called a habitat suitability index (HSI). The value of HSI, which is the same as fitness in other population-based algorithms, depends on many features of the habitat, where a high and low HSI of a habitat represent a good and bad performance on the optimization problem, respectively. In fact, in any optimization problem, the objective function is considered as HSI. The BBO algorithm considers two main operators: migration operator, which includes both emigration and immigration; and the mutation operator.

BBO Algorithm Formation

In the following, the characteristics of the BBO algorithm are explained:

Habitat Modification and Mutation Probabilities along with Elitism Parameter

Habitat modification probability is similar to crossover probability in GA. Mutation probability and elitism parameter are the same as in GA. For BBO, the mutation probability, inversely proportional to the solution probability, is defined as follows:

$$m_i = m_{max}\left(1 - \frac{i}{P_{max}}\right) \tag{13}$$

Where m_{max} is the user-defined maximum mutation probability, $P_{max} = argmaxP_i$, and P_i is the solution probability. With probability P_{mod}, each solution is modified based on other solutions. The mutation is a probabilistic operator that is used randomly for modifying solution's SIV based on its prior probability of existence. Mutation tends to increase diversity among the population. Therefore, the mutation probability obtained by Equation (13) gives a chance of not only improving for low HSI solutions but also more improving for high HSI solutions. The steady-state value for the solution probability is given by follows:

$$P_i = \frac{v_i}{\sum_{i=1}^{n+1} v_i} \quad and\, v = \left[v_1, v_2, \cdots, v_{(n+1)}\right]^T \tag{14}$$

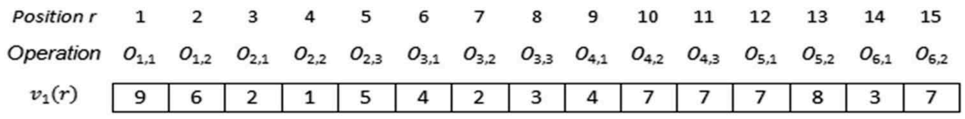
Where v and v_i are the functions of the population size. v and v_i are defined by the next equations;

$$v_i = \begin{cases} \frac{n!}{(n+1-i)!(i-1)!}, & i = 1, 2, \ldots, i' \\ v_{n+2-i}, & i = i' + 1, \ldots, n+1 \end{cases} \tag{15}$$

Where i' is the smallest integer that is greater than or equal to $(n+1)/2$.

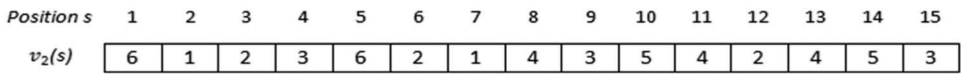
Solution Representation and Decoding of Habitats

The initial population of habitats is randomly generated. In the VCMS, each habitat is a combination of two decisions; the operation assignment of the jobs on machines as well as the processing sequence of operations on the machines. With respect to the solution representation developed by Gao, Sun, and Gen (2008), a combination of machine assignment vector, $v_1(r)$, and operation sequence vector, $v_2(s)$, represents a solution. An example solution related to six jobs ($n = 6$) including 15 operations ($\sum_{j \in N} h_j = 15$) and nine machines ($m = 9$) including three machine types ($\mathcal{L} = 3$, $\sum_{i \in I} v_i = 9$) is depicted by Figures 2 and 3. For simplicity, the machine and job operation indices are shown without the machine type and operation number, respectively.



Position r	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Operation	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$	$O_{3,3}$	$O_{4,1}$	$O_{4,2}$	$O_{4,3}$	$O_{5,1}$	$O_{5,2}$	$O_{6,1}$	$O_{6,2}$
$v_1(r)$	9	6	2	1	5	4	2	3	4	7	7	7	8	3	7

Figure 2. Illustration for permutation type machine assignment vector.



Position s	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$v_2(s)$	6	1	2	3	6	2	1	4	3	5	4	2	4	5	3

Figure 3. Illustration for permutation type operation sequence vector.

Figure 2 depicts a machine assignment vector. In each machine assignment vector, $v_1(r)$ represents the machine assigned for the operation that is ordered in the position r. And in each operation sequence vector depicted in Figure 3, $v_2(s)$ represents the number of jobs that are ordered in the priority s. Each job $j \in N$ emerges h_j times to represent its h_j ordered operations.

To increase the comprehension of the VCMS concept, assume that the scheduling horizon is divided into time periods, so when the first operation of any job starts in a period, its next operation must start in the next period and so forth. Apart from this, operations can only be started at the beginning of the time period even if preceding operations finished early. Because of these precedence constraints among operations of the same job, idle time may exist between operations on a machine. Therefore, it is necessary to process decoding habitats into the VCMS schedules.

When the operation $O_{j,h}$ is scheduled on machine s_{iv}, the aim is to search for the earliest time interval on the machine s_{iv} with respect to the precedence constraint. This being the case, the operations that have already been scheduled on machine s_{iv} must be evaluated from left to right to find the earliest time interval $\left[t^B_{s_{iv}}, t^E_{s_{iv}}\right]$ for $O_{j,h}$, beginning from $t^B_{s_{iv}}$ and ending at $t^E_{s_{iv}}$ on machine s_{iv}. If there exists such an available time interval for $O_{j,h}$, it is allocated there; otherwise, it is allocated at the end of the machine s_{iv}. This decoding process for the habitat, represented by Figures 2 and 3, is depicted schematically by a Gantt chart in Figure 4.

Calculate the Immigration Rate (λ_i) and Emigration Rate (μ_j)

These migration rates (i.e., immigration and emigration rates) of a habitat can be modeled as Figure 5 and variation of immigration and emigration rates for different species abundance in a habitat are functions of the fitness or *HSI* of the solution which can be calculated as follows:

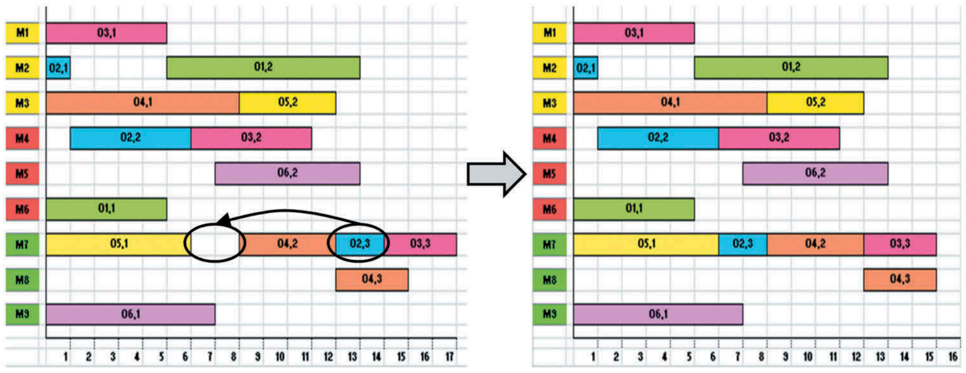


Figure 4. Decoding Gantt chart for the habitat related to Figures 2 and 3.

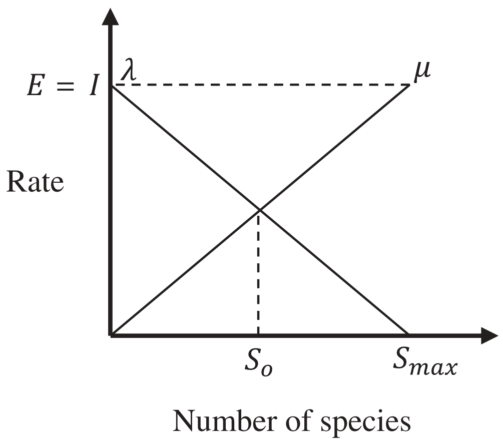


Figure 5. Variation of migration rates for different species abundance in a habitat.

$$\lambda_i = I\left(1 - \frac{k(i)}{n}\right) \tag{16}$$

$$\mu_j = E\left(\frac{k(i)}{n}\right) \tag{17}$$

Where I and E are the maximum possible immigration and emigration rates, respectively. $k(i)$ is the fitness rank of the i^{th} individual (1 is worst and n is best) and n is the number of candidate solutions in the population. I and E are often set equal to 1 or slightly less than 1.

S_{max} defines the largest number of species in each habitat, while S_o defines equilibrium position which these two migration rates are equal. A habitat with high HSI is a good solution, while the one with low HSI is a poor solution. High HSI solutions tend to share their features with low HSI solutions (emigration process) so that low HSI solutions accept new features from high HSI solutions (immigration process). Therefore, migration operators which are emigration and immigration are used to improve and evolve a solution to the optimization problem. Each habitat has an immigration rate λ_i and emigration rate μ_j for itself. The habitats with high HSI have a high emigration rate, while the habitats with low HSI have a high immigration rate. Figure 5 illustrates the relationship between species abundance, immigration rate λ_i, and emigration rate μ_j in a single habitat. Both the immigration rate λ_i and the emigration rate μ_j are a function of the number of species in the habitat, i.e., λ_i decreases and μ_j increases as the number of species are increased. Therefore, according to the above argument, it is expected that a high-*HSI* solution has high μ_j and low λ_i, while a low-*HSI* solution has low μ_j and high λ_i. It means that *SIVs* of a high-*HSI* solution tend to emigrate to low-*HSI* solutions.

Operator Structure

An immigration habitat H_i is selected probabilistically based on the immigration rate λ_i and then select an emigration habitat H_j for emigrating to H_i. In this paper, the roulette wheel selection method is used in order to select the migration habitats and solution features in the developed BBO algorithm.

Migration Operator

Migrate randomly selected *SIVs* based on the selected habitats in the previous step. The probability that the solution H_i is selected as the immigrating/emigrating habitat is proportional to its immigration/emigration rate λ_i/μ_j. Migration operator can be expressed as:

$$H_i(SIV) \leftarrow H_j(SIV)$$

This equation means that a feature of the solution H_i is replaced by a feature from solution H_j. In fact, in the migration process, each solution H_i should be modified by sharing features between other solutions H_j. This procedure is performed in two vectors: machine assignment and operation sequencing. Part (a) of Figure 6 shows two habitats which are selected probabilistically and two *SIVs* which would be replaced. And part (b) of Figure 6 shows these two habitats after performing the migration operator. The migration operator is performed similarly to the crossover operator in GA.

Likewise, part (a) of Figure 7 shows two operation sequencing vectors with selected SIVs, while part (b) of Figure 7 shows these two operation sequencing vectors after performing the migration operator. Since the number of jobs will be changed after implementing the migration operator, a modification process should be applied to this vector. In the modification process, firstly, the job number replaced by an *SIV* is selected out of an emigration habitat. Next, the earliest job existed after this position should be

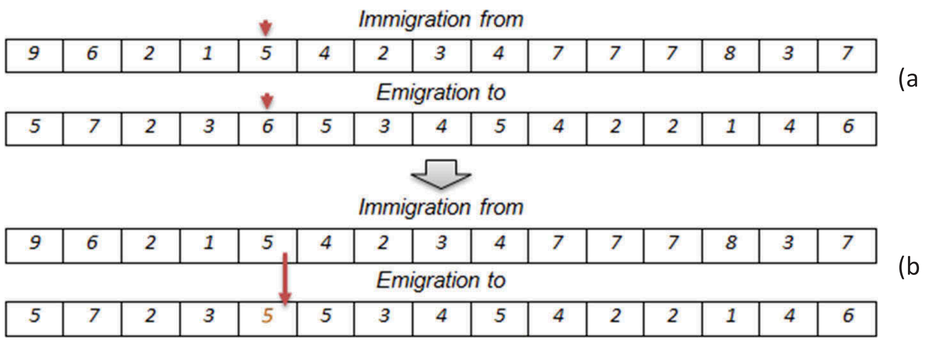


Figure 6. Illustration for migration operator on machine assignment vector.

found. And then it is changed to the job number which was selected in the immigration habitat. Part (c) of Figure 7 illustrates this modification process.

The proposed migration framework is shown in Figure 8 as follows:

Mutation Operator on Habitats

Probabilistically, the mutation is performed based on the mutation probability for each habitat. Mutation is a probabilistic operator that randomly modifies a solution's *SIV* on its prior probability of existence P_i. In this paper, the *SIV*-based mutation is used. In other words, for a machine assignment vector, a mutation operator decides whether an *SIV* should be selected for mutation with a certain probability, and then a newly available machine is assigned for the operation indicated by the selected *SIV*. Likewise, for an operation sequence vector, it randomly decides whether to mutate an *SIV* on a certain probability. If an *SIV* is to be mutated, then $v_2(s-1)$ and $v_2(s)$ are swapped.

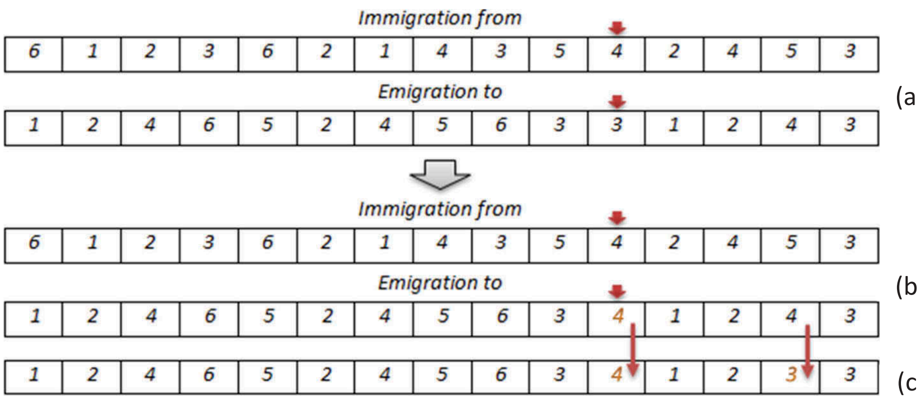


Figure 7. Illustration for migration operator on operation sequence vector.

```
Select $H_i$ with probability $\alpha\ \lambda_i$
If $H_i$ is selected
    For $j = 1$ to $n$
        select $H_j$ with probability $\alpha\ \mu_i$
        If $H_j$ is selected
            Randomly select an SIV $\sigma$ from $H_j$
            Replace a random SIV in $H_i$ with $\sigma$
        end
    end
end
```

Figure 8. Illustration for coding migration operator in the BBO algorithm.

Figures 9 and 10 show a simple example of implementing a mutation operator on a machine assignment and operation sequence vectors, respectively.

The proposed mutation framework is shown in Figure 11.

It is worth noting that in the BBO algorithm, the next generation is produced by immigrating solution features to other habitats and receiving solution features by emigrating from other habitats. The mutation is performed for the whole population in a manner similar to the mutation in GA. The steps of the developed BBO algorithm is described with the help of the pseudo-code shown in Figure 12.

Hybridization of the BBO and GA Algorithms

A combination of two population-based algorithms, i.e., BBO and GA, is considered as a hybrid algorithm. The variables and parameters of the hybrid algorithm are the same as basic GA and BBO. In this algorithm, firstly, the BBO algorithm is conducted for a problem until the stop criterion is satisfied. Then, the best-recorded solution along with a new population again will be conducted this time by GA. This proposed algorithm structure allows each solution to be conducted with two optimization algorithm sequentially.

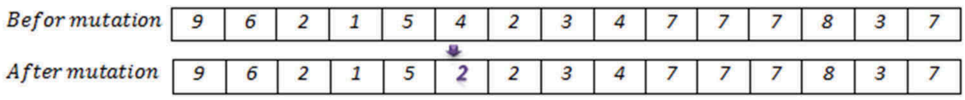


Figure 9. Illustration for mutation operator on machine assignment vector.

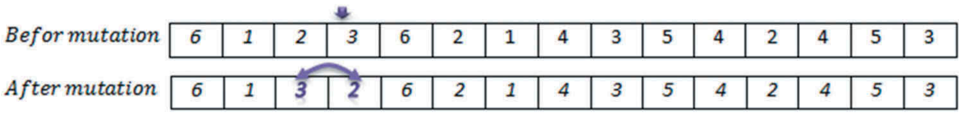


Figure 10. Illustration for mutation operator on operation sequence vector.

```
For $j = 1$ to $m$
    Use $\lambda_i$ and $\mu_i$ to compute the pribabolity $P_i$
    Select SIV $H_i(j)$ with probability $\alpha P_i$
    If $H_i(j)$ is selected
        Replace $H_i(j)$ with a randomly generated SIV
    end
end
```

Figure 11. Illustration for coding mutation operator in the BBO algorithm.

```
Initialize $m_{\max}$ , maximum migration rates $E$ and $I$, number of iteration, and an elitism parameter;
Initialize a random set of habitats (population) $H_1, H_2, \ldots, H_n$ ;
Compute the HSI for each habitat;
For $i$=1 to *number of iteration*
    Sort the population according to their HSI
    For $j$=1 to $n$ do
        Calculate the immigration rate $\lambda_i$ and the emigration rate $\mu_i$ for each habitat;
    End-For
    Select non-elite $H_i$ with probability proportional to $\lambda_i$
    If $H_i$ is selected
        For $j$=1 to $n$ do
            Select non-elite $H_j$ with probability proportional to $\mu_i$
            If $H_j$ is selected
                Execute migration operator
            End-If
        End-For
    End-If
    Calculate $p_j$ and mutation rate $m_j$ for each habitat
    Select a habitat with probability proportional to $m_i$
    If non-elite $H_i$ is selected
        Execute mutation operator
    End-If
    For $i$=1 to $n$ do
        Evaluate habitats in the population
    End-For
End-For
```

Figure 12. Pseudo-code for the developed BBO algorithm.

Higher solution quality and low computational time are as a result of the hybridization of these two population-based algorithms compared to a basic BBO or GA. In the following, the crossover and mutation operators of GA, as the important futures of GA, are explained. These operators are implemented by both the basic GA and GA accompanied by basic BBO.

GA is a robust adaptive and stochastic evolutionary algorithm. Two types of operators are participated in GA: mutation which creates new individuals by making changes in a single individual; and crossover which creates new individuals by combining two or more individuals. Since a large amount of population is updated with the help of the crossover operator, the order-crossover-related move (OX) is applied for implementing the crossover operator. At first, two distinct integers, χ_1 and χ_2, randomly generate between 2 and ($\sum_{j\in N} h_j - 1$) as cutting points, where $\sum_{j\in N} h_j$ is the total number of operations belonging to the jobs. The orders between $\chi_1 - 1$ and $\chi_2 + 1$

belonging to the first parent are inserted into the same position (of the first parent) on the first offspring. Likewise, the orders between $\chi_1 - 1$ and $\chi_2 + 1$ belonging to the second parent are inserted into the same position (of the second parent) on the second offspring. Finally, the other orders (i.e., the orders before and after χ_1 and χ_2, respectively) of the first parent are put in the empty positions of the second offspring and vice versa. In this step, the iterative order should be changed to the one which is not appeared in the offspring, if an order is repeated.

The mutation operator in GA is usually used for improving the diversity of the algorithm. The swap-related move is applied for mutation operator. In other words, two genes are randomly chosen and their positions are exchanged.

BBO Algorithm with Restart Phase

The BBO/RF algorithm represents the BBO algorithm with a restart on each iteration of the search algorithm. In other words, in the first start of each iteration, the BBO algorithm is performed on a problem until a stop criterion is satisfied. Then, in the second restart, the BBO algorithm applies the best solution along with the previous population as its initial population. The aim of this restart phase in each iteration is that the best solution obtained in the first start of BBO would allow to improve more and enhance its quality in a shorter time.

All types of the BBO algorithms must be continued until they reach the number of pre-determined iterations. In this paper, the upper bound of the number of iteration is set to 800. For the evaluation of the proposed algorithms, their performance is examined over benchmark problems from the literature (Kessen, Sanchoy, and Gungor 2010) and compared with the last published GA of the literature in terms of the efficiency and effectiveness.

Computational Results

All algorithms are allowed to run on identical computers with Intel(R) Core(TM) 2 Duo, 2.00 GHz processors & 4.00 GB RAM in order to conduct the numerical experiments. Apart from this, all algorithms are programmed in MATLAB 7.10.0 (R2010b). Algorithm solutions are compared based on a set of test problems. With regard to the data generation mechanism in the literature, for each test problem, the processing time of jobs on each machine corresponds to a uniform distribution, i.e., $P_{j,h} = unif[2,\ 10]$. The distance between each pair of machines belonging to different machine types corresponds to a uniform distribution, i.e., $D_{s_{iv},s_{i'v'},j,h} = unif[5,\ 40]$. The weights attributed to the makespan and the total traveling distance in the objective function are considered as $\alpha = 0.95$ and $\beta = 0.05$, respectively.

The parameters employed in the BBO and GA algorithms are selected experimentally to obtain a satisfactory solution quality in an acceptable time span. In relation to both BBO and GA, the population size is 100 and the maximum iterations of the algorithms are 800. The crossover rate and mutation rate in GA are 0.45 and 0.30, respectively, while the migration rate and the mutation rate in BBO are 0.90 and 0.10, respectively. Due to the deterministic nature of the BBO and GA algorithms, they are conducted with multiple runs on each problem to get coherent results. In other words, with the help of Taguchi design and several scenarios for each parameter, the algorithms are run five times for each combination of parameters related to a problem. Then, after each run, those obtained results in terms of the best solution which has the minimum cost and computational time to get the best solution are recorded.

Figures 13-15 demonstrate the convergence speed and best value of the objective function of basic GA (Alg_{GA}), BBO (Alg_{BBO}), the Hybrid algorithm ($Alg_{BBO/GA}$), and the BBO/RF algorithm ($Alg_{BBO/RF}$) with regard to computational time, in large, medium, and small size of a sample problem, respectively. The performance of algorithms is compared with solid and dotted lines related to the best and the mean values, respectively. Figures suggest a good quality solution with the fast convergence speed for the BBO algorithms compared to Alg_{GA}. Apart from this, the hybrid algorithm and $Alg_{BBO/RF}$ present the better results compared to Alg_{GA} and Alg_{BBO}.

All algorithms are compared with each other based on their effectiveness and efficiency. Algorithm effectiveness and efficiency are determined in terms of the best solution and related computational time, respectively. With the help of an individual set of test problems, the performance of developed algorithms is evaluated based on the best solution and related computational time obtained by all algorithms for each test problem.

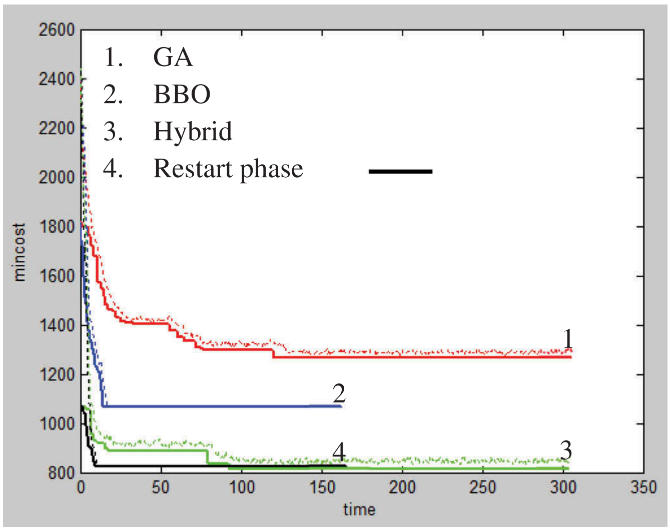


Figure 13. Illustration for the convergence speed and best solution in a large problem.

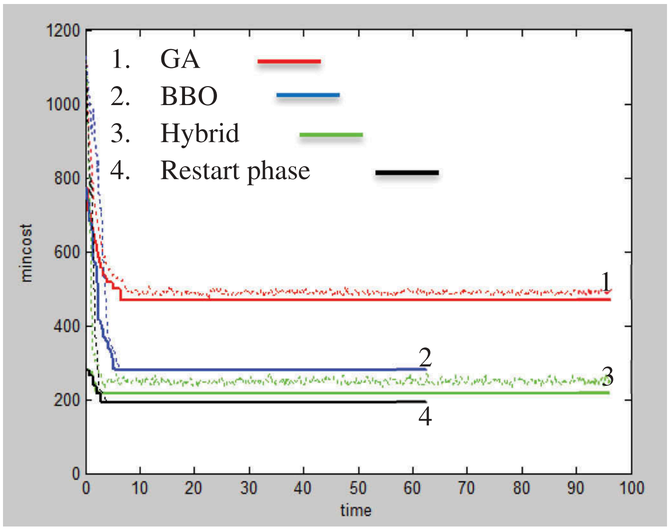


Figure 14. Illustration for the convergence speed and best solution in a medium problem.

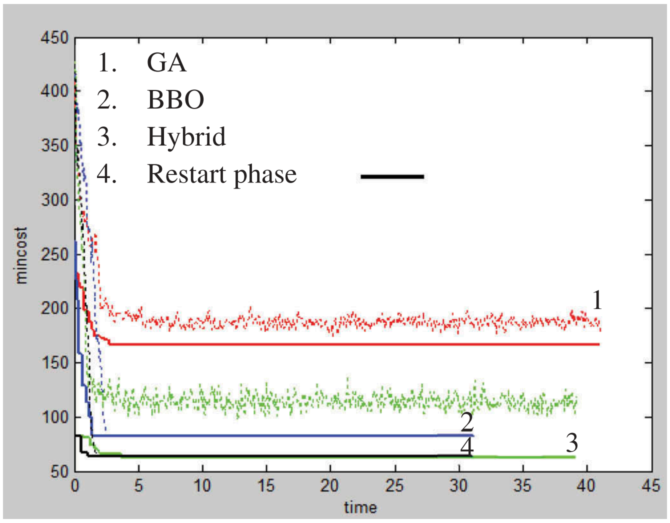


Figure 15. Illustration for the convergence speed and best solution in a small problem.

Figures 16 and 17 illustrate the interval plot for *PD* value and computational time of algorithms, respectively. In the effectiveness point of view, the performance of algorithms sorts from the best to worst as $Alg_{BBO/RF} \rightarrow Alg_{BBO/GA} \rightarrow Alg_{BBO} \rightarrow Alg_{GA}$, while in the efficiency point of view, they rank as $Alg_{BBO/RF} \rightarrow Alg_{BBO} \rightarrow Alg_{BBO/GA} \rightarrow Alg_{GA}$. It can be concluded that not only GA has a significant difference with the BBO algorithms, but also the BBO algorithms, particularly $Alg_{BBO/RF}$, are more effective and converge faster than GA.

Table 1. Performance of all developed algorithms compared to the best solution & CPU time.

# of jobs	# of M|Cs	Alg_{GA}		Alg_{BBO}		$Alg_{BBO/GA}$		$Alg_{BBO/RF}$		Alg_{Best}		UB_{GA} vs UB_{Best}	UB_{BBO} vs UB_{Best}	$UB_{BBO/GA}$ vs UB_{Best}	$UB_{BBO/RF}$ vs UB_{Best}
n	*m*	*UB*	*CPU*	*UB*	*CPU*	*UB*	*CPU*	*UB*	*CPU*	*UB*	*CPU*	184.8%	21.7%	21.7%	0.0%
4	3	131	4.56	56	7.71	56	0.89	46	1.08	46	1.08	153.5%	0.0%	0.0%	0.0%
4	5	109	0.42	43	0.89	43	1.45	43	0.87	43	0.87	220.0%	126.7%	126.7%	0.0%
5	2	96	1.50	68	17.15	68	3.14	30	5.79	30	5.79	95.9%	8.2%	6.1%	0.0%
5	4	96	2.46	53	22.56	52	1.67	49	1.70	49	1.70	85.3%	0.0%	0.0%	0.0%
6	3	252	0.94	136	2.53	136	2.14	136	3.68	136	3.68	62.7%	30.7%	18.7%	0.0%
6	5	122	31.50	98	5.57	89	2.59	75	4.43	75	4.43	79.3%	25.0%	3.3%	0.0%
7	4	165	1.79	115	2.40	95	4.12	92	1.67	92	1.67	60.3%	27.3%	6.6%	0.0%
7	5	194	32.46	154	5.20	129	33.32	121	6.78	121	6.78	296.2%	20.5%	0.0%	20.5%
8	4	309	5.18	94	68.39	78	56.35	94	54.74	78	56.35	147.0%	22.9%	19.3%	0.0%
8	6	205	19.99	102	69.17	99	14.58	83	13.28	83	13.28	147.3%	26.0%	0.0%	22.0%
9	3	371	5.04	189	2.53	150	32.84	183	11.84	150	32.84	149.5%	45.8%	35.5%	0.0%
9	5	267	3.40	156	3.06	145	1.90	107	2.95	107	2.95	133.7%	26.1%	26.1%	0.0%
10	3	215	55.62	116	26.52	116	17.55	92	20.63	92	20.63	75.3%	7.5%	8.4%	0.0%
10	5	419	27.55	257	56.32	259	44.73	239	49.28	239	49.28	58.6%	22.0%	10.8%	0.0%
10	6	498	16.96	383	12.97	348	77.31	314	5.41	314	5.41	30.3%	21.7%	0.0%	6.3%
11	3	331	82.09	309	58.71	254	58.65	270	42.22	254	58.65	111.7%	26.4%	6.4%	0.0%
11	4	561	6.51	335	60.69	282	67.46	265	6.96	265	6.96	59.3%	41.5%	8.1%	0.0%
11	6	411	43.43	365	23.34	279	41.36	258	6.35	258	6.35	201.5%	13.2%	2.2%	0.0%
12	4	410	14.55	154	27.18	139	79.09	136	53.73	136	53.73	79.0%	27.2%	0.0%	21.0%
12	6	666	68.53	473	15.58	372	11.17	450	4.85	372	11.17	112.6%	24.7%	16.6%	0.0%
13	3	525	61.76	308	62.70	288	81.76	247	4.13	247	4.13	147.6%	65.6%	21.1%	0.0%
14	5	562	71.65	376	52.12	275	84.76	227	67.38	227	67.38	188.7%	48.6%	36.1%	0.0%
15	3	921	62.71	474	130.98	434	35.23	319	45.07	319	45.07	51.5%	36.1%	9.3%	0.0%
17	3	571	56.77	513	46.88	412	24.98	377	39.92	377	39.92	79.9%	73.9%	0.0%	21.8%
18	4	759	37.64	734	63.62	422	99.64	514	96.75	422	99.64	48.3%	36.6%	5.1%	0.0%
20	5	899	122.82	828	71.81	637	85.22	606	117.09	606	117.09	21.5%	16.9%	0.0%	3.6%
22	4	915	138.37	880	110.85	753	137.51	780	156.50	753	137.51	24.5%	24.1%	0.0%	3.1%
23	3	935	147.54	932	258.14	751	212.85	774	122.26	751	212.85	37.2%	36.0%	5.7%	0.0%
25	4	1063	245.25	1054	253.44	819	248.24	775	240.13	775	240.13	59.5%	46.8%	6.7%	0.0%
25	5	1311	57.78	1207	163.94	877	103.46	822	119.99	822	119.99	184.8%	21.7%	21.7%	0.0%
											Average:	106.7%	31.7%	13.3%	3.3%

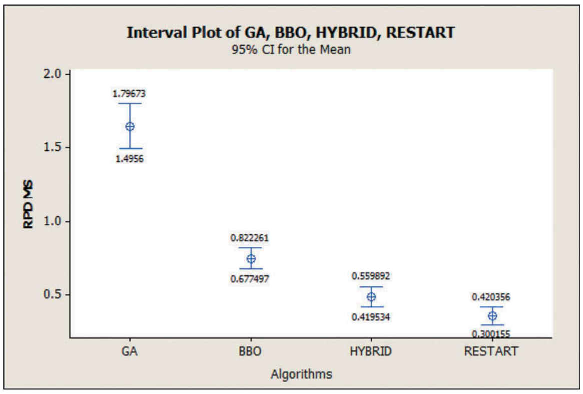


Figure 16. The interval plot for the best solution.

Table 1 shows the performance of all algorithms are evaluated based on the best solution and related computational time obtained by all algorithms for each test problem. The results indicate that the solutions obtained by $Alg_{BBO/RF}$ are very effective and converges faster compared to those obtained by Alg_{GA} and other BBO algorithm. The relative percentage deviation (RPD) is applied to evaluate the performance of the developed algorithms by each other. RPD Value represents the gap between each solution obtained by an algorithm and the best solution obtained by all algorithms. RPD is calculated as follows:

$$RPD = \frac{Alg_i - \ Alg_{Best}}{Alg_{Best}} \times 100, \quad \forall i \in \{GA, BBO, BBO/GA, BBO/RF\}, \quad (18)$$

where Alg_i is a solution obtained by i^{th} algorithm and Alg_{Best} is the best solution obtained by all algorithms. The average of RPD values shows $Alg_{BBO/RF}$ and Alg_{GA} have the minimum deviation (3.3%) and maximum deviation (106.7%) from the best solutions, respectively. Also, $Alg_{BBO/RF}$ has

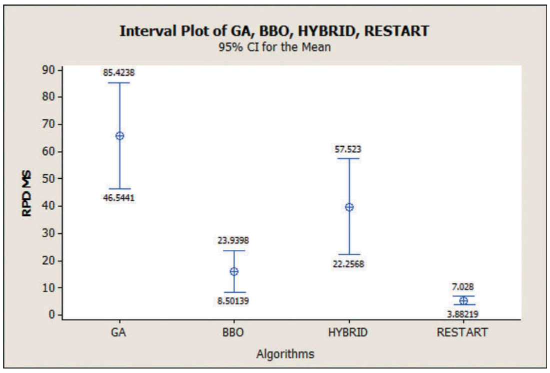


Figure 17. The interval plot for the CPU time.

the best performance amongst the developed BBO algorithms and $Alg_{BBO/GA}$ represents the better performance compared to the basic BBO algorithm, i.e., Alg_{BBO}. The principal result of a paired t-test performed to compare different search algorithms based on solution and computation time are shown in Tables 2 and 3, respectively.

Based on P_{value} at significant level of 5% for each comparison of solutions, it can be concluded that there is a significant difference between the results of these algorithms except $Alg_{BBO/GA}$ and $Alg_{BBO/RF}$. Since the average objective function value of $Alg_{BBO/RF}$ is less than all other search algorithms, it can be concluded that $Alg_{BBO/RF}$ provides better solutions for the proposed research problem. Likewise, $Alg_{BBO/GA}$ presents a better performance compared to Alg_{BBO} and Alg_{GA}. Based on P_{value} at significant level of 5% for each comparison of computational times, there is no difference between developed search algorithms except $Alg_{BBO/GA}$ and $Alg_{BBO/RF}$.

Performance of $Alg_{BBO/RF}$ vs. the state-of-the-Art Algorithm

In order to show the superiority performance of the best BBO algorithm, i.e., $Alg_{BBO/RF}$ a benchmark of the VCMS scheduling problems with the same

Table 2. Paired t-test between solutions of developed search algorithms.

	Paired differences							
				95% confidence interval of the difference				
Pair	Mean	Std. deviation	Std. error mean	Lower	Upper	*t*	*df*	Sig. (2-tailed)
$Alg_{GA} - Alg_{BBO}$	110.900	96.926	17.696	73.862	147.938	6.267	29	< 0.00001
$Alg_{GA} - Alg_{BBO/GA}$	181.067	116.160	21.208	136.679	225.455	8.538	29	< 0.00001
$Alg_{GA} - Alg_{BBO/RF}$	192.167	128.409	23.444	143.098	241.235	8.197	29	< 0.00001
$Alg_{BBO} - Alg_{BBO/GA}$	70.167	92.744	16.933	34.727	105.607	4.144	29	0.00027
$Alg_{BBO} - Alg_{BBO/RF}$	81.267	94.671	17.285	45.090	117.443	4.702	29	0.000058
$Alg_{BBO/GA} - Alg_{BBO/RF}$	11.100	39.155	7.149	−3.862	26.062	1.553	29	0.13127

Table 3. Paired t-test between computational times of developed search algorithms.

	Paired differences							
				95% confidence interval of the difference				
Pair	Mean	Std. deviation	Std. error mean	Lower	Upper	*t*	*df*	Sig. (2-tailed)
$Alg_{GA} - Alg_{BBO}$	−9.205	40.360	7.369	−24.628	6.218	−1.249	29	0.221653
$Alg_{GA} - Alg_{BBO/GA}$	−7.972	33.959	6.200	−20.949	5.005	−1.286	29	0.208617
$Alg_{GA} - Alg_{BBO/RF}$	3.977	30.065	5.489	−7.512	15.466	0.725	29	0.474262
$Alg_{BBO} - Alg_{BBO/GA}$	1.233	33.178	6.057	−11.445	13.911	0.204	29	0.839778
$Alg_{BBO} - Alg_{BBO/RF}$	13.182	37.187	6.789	−1.028	27.392	1.942	29	0.61907
$Alg_{BBO/GA} - Alg_{BBO/RF}$	11.949	29.068	5.307	0.841	23.057	2.252	29	0.032065

objective functions is considered. This benchmark is to compare the two algorithms, i.e. $Alg_{BBO/RF}$ and the GA algorithm in the literature (Kessen, Sanchoy, and Gungor 2010), i.e., Alg^{*}_{GA}. It consists of 30 available test problems developed by Kessen, Sanchoy, and Gungor (2010). Similar to the procedure mentioned in the literature (Kessen, Sanchoy, and Gungor 2010) and in order to keep consistency in the comparison, we conduct multiple runs (five times) on each problem to get coherent results and, consequently, the best result is reported to each problem. Table 4 shows the comparative results obtained for $Alg_{BBO/RF}$ and Alg^{*}_{GA}.

Although $Alg_{BBO/RF}$ and Alg^{*}_{GA} are coded by two different machines with very close configurations, $Alg_{BBO/RF}$ reduces the computational time of Alg^{*}_{GA} up to 7.3% in average. The deviation between the two aforementioned algorithms is calculated for each test problem as follows:

Table 4. Performance of $Alg_{BBO/RF}$ with respect to the benchmark in the literature (Kessen, Sanchoy, and Gungor 2010).

	# of jobs	# of M|Cs	Alg^{*}_{GA}		$Alg_{BBO/RF}$		Percentage deviation
Test Problem	*n*	*m*	*OFV*	*CPU**	*OFV*	*CPU*	$\Delta\%$
1	4	3	523	<1	**523**	1.08	0.00%
2	4	3	420	<1	**420**	0.99	0.00%
3	4	4	757	<1	**757**	0.83	0.00%
4	6	2	836	<1	**836**	1.24	0.00%
5	6	3	706	2	**706**	3.70	0.00%
6	6	3	497	2	**497**	2.93	0.00%
7	6	3	549	2	**549**	2.69	0.00%
8	8	3	906	4	**906**	4.02	0.00%
9	8	3	1047	5	**1047**	3.99	0.00%
10	8	4	967	7	**967**	34.54	0.00%
11	8	4	1084	6	**1084**	23.12	0.00%
12	8	4	1207	9	**1207**	13.45	0.00%
13	10	3	1119	10	1121	19.55	−0.18%
14	10	3	1299	13	**1299**	13.31	0.00%
15	10	4	1227	13	1227	11.10	0.00%
16	10	5	1387	16	**1387**	41.45	0.00%
17	10	6	1666	20	1631	7.44	2.10%
18	12	2	1089	13	1067	8.34	2.02%
19	12	3	1442	17	1425	10.01	1.18%
20	12	3	672	13	672	13.78	0.00%
21	12	4	1841	19	1803	41.73	2.06%
22	12	6	1572	36	1528	11.97	2.80%
23	12	6	1710	38	1715	13.23	−0.29%
24	15	3	1669	33	1643	40.07	1.56%
25	15	3	1061	29	1059	23.76	0.19%
26	15	4	2247	40	2229	39.21	0.80%
27	15	4	1549	36	1521	43.01	1.81%
28	15	5	1821	54	1821	45.23	0.00%
29	20	4	1768	70	1719	53.56	2.77%
30	20	5	2714	194	2671	123.86	1.58%

*The average CPU time of all five runs for each test problem reported in the literature (Kessen, Sanchoy, and Gungor 2010).

$$\Delta\% = \frac{OFV_{Alg^*_{GA}} - OFV_{Alg_{BBO/RF}}}{OFV_{Alg^*_{GA}}} \times 100$$

where OFV_i indicate the best objective function value obtained by algorithm i over five runs. Similar to Alg^*_{GA}, $Alg_{BBO/RF}$ was able to solve 14 out of 30 test problems, optimally, shown by bold values in Table 4. For the rest of the problems, $Alg_{BBO/RF}$ was able to find the same or better results compared to Alg^*_{GA} except for 2 out of 16 problems (i.e., problems 13 and 23). The improvement percentages in the objective function value in $Alg_{BBO/RF}$ presented in the last column of Table 4 show up to 2.8% improvement in the objective function value obtained from Alg^*_{GA}. As a result, $Alg_{BBO/RF}$ guarantees to give the solution that is at least as good as the solution obtained from Alg^*_{GA}.

Conclusion and Future Works

This paper has presented some effective and efficient schedules for the VCMS problems with the help of the BBO algorithm. In the VMC configuration, machines with different processing abilities are located in close proximity to enhance the entire system's ability against the other changes, thereby providing more than one alternative routes for the jobs. Makespan is one of the most popular performance criteria because it reflects the utilization of machines. Separation of machines with similar processing abilities raises the issue of total traveling distance. Heavily, utilized shortest routes for each job affect the makespan value adversely. This being the case, the objective function is to minimize the weighted sum of the makespan and total traveling distance.

Since the problem is among NP-hard problems, the computational time dramatically increases from small-size to medium- and large-size problems. Hence, meta-heuristic techniques must be applied to find good quality solutions. Therefore, several meta-heuristics in terms of the GA and BBO algorithm are developed for the VCMS scheduling problem. The performance of developed algorithms, i.e., the basic GA algorithm, the basic BBO algorithm, the hybridization of BBO and GA, and the BBO/RF algorithm, are compared with the help of 30 test problems. Apart from this, the performance of the best-developed algorithm is compared to the performance of the state-of-the-art algorithm in the literature, with the help of 30 available test problems developed in the literature. The results clearly show that the BBO/RF algorithm has the best performance based on efficiency and effectiveness of solutions between amongst algorithms. In addition, the BBO/RF algorithm and a basic GA have the minimum deviation (3.3%) and maximum deviation (106.7%) from the best solutions obtained by all algorithms, respectively, and all developed BBO algorithms present superior performance compared to GA. Finally, the BBO/RF algorithm presents a superior performance compared to the state-of-the-art algorithm in the literature.

Some directions for future research are to consider the scheduling constraints such as machine failure, sequence-dependent setup times, workload balancing, batch splitting. Also, incorporation of other effective metaheuristic algorithms, multi-objective algorithms, and learning procedures can be considered in the VCMS scheduling problems. Finally, a fuzzy version of the BBO algorithm through fuzzy fitness or fuzzy processing time and handling time can be developed in the future research.

References

Aalaei, A., and Davoudpour, H. 2016. Revised multi-choice goal programming for incorporated dynamic virtual cellular manufacturing into supply chain management: A case study. *Engineering Applications of Artificial Intelligence* 47:3–15. doi:10.1016/j.engappai.2015.04.005.

Ahkioon, S., A. A. Bulgak, and T. Bektas. 2009. Integrated cellular manufacturing systems design with production planning and dynamic system reconfiguration. *European Journal of Operational Research* 192:414–28. doi:10.1016/j.ejor.2007.09.023.

Balakrishnan, J., and C. H. Cheng. 2005. Dynamic cellular manufacturing under multi period planning horizons. *Journal of Manufacturing Technology Management* 16:516–30. doi:10.1108/17410380510600491.

Baykasoglu, A., and L. Gorkemli. 2017. Dynamic virtual manufacturing through agent-based modelling. *International Journal of Computer Integrated Manufacturing*. doi:10.1080/0951192X.2016.1187294.

Defersha, F., and M. Chen. 2006. A comprehensive mathematical model for the design of cellular manufacturing systems. *International Journal of Production Economics* 103:767–83. doi:10.1016/j.ijpe.2005.10.008.

Drolet, J., Y. Marcoux, and G. Abdulnour. 2008. Simulation-based performance comparison between dynamic cells, classical cells, and job shops: A case study. *International Journal of Production Research* 46 (2):509–36. doi:10.1080/00207540601138312.

Du, D. 2009. Biogeography-based optimization: Synergies with evolutionary strategies, immigration refusal and Kalman filters. Master Thesis, Cleveland State University.

Fu, L., and T. Murata. 2016. Configuration design of virtual cellular manufacturing system with batch splitting operations. 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), 1010–15. Kumamoto.

Gao, J., L. Sun, and M. Gen. 2008. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers and Operations Research* 35 (9):2892–907. doi:10.1016/j.cor.2007.01.001.

Garg, H. 2014. Solving structural engineering design optimization problems using an artificial bee colony algorithm. *Journal of Industrial and Management Optimization* 10 (3):777–94. doi:10.3934/jimo.2014.10.777.

Garg, H. 2015a. A hybrid GA-GSA algorithm for optimizing the performance of an industrial system by utilizing uncertain data. In *Handbook of research on artificial intelligence techniques and algorithms*, ed. P. Vasant, 620–54. IGI Global. doi:10.4018/978-1-4666-7258-1.ch020.

Garg, H. 2015b. An efficient biogeography based optimization algorithm for solving reliability optimization problems. *Swarm and Evolutionary Computation* 24:1–10. doi:10.1016/j.swevo.2015.05.001.

Garg, H. 2016a. A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation* 274:292–305. doi:10.1016/j.amc.2015.11.001.

Garg, H. 2016b. A new approach for solving fuzzy differential equations using Runga-Kutta and Biogeography-based optimization. *Journal of Intelligent and Fuzzy Systems* 30:2417–29. doi:10.3233/IFS-152010.

Hamedi, M., and G. Esmaeilian. 2015. Functional and distributed layouts and their effectiveness on capability-based virtual cellular manufacturing systems performance. 2015 International Conference on Industrial Engineering and Operations Management (IEOM), 1–8. Dubai.

Kannan, V. R. 1998. Analyzing the trade-off between efficiency and flexibility in cellular manufacturing systems. *Production Planning and Control* 9 (6):572–79. doi:10.1080/095372898233821.

Kannan, V. R., and S. Ghosh. 1996a. cellular manufacturing using virtual cells. *International Journal of Operations and Production Management* 16 (5):99. doi:10.1108/01443579610113979.

Kannan, V. R., and S. Ghosh. 1996b. A virtual cellular manufacturing approach to batch production. *Decision Sciences* 27 (3):519–39. doi:10.1111/deci.1996.27.issue-3.

Kessen, S. E., K. Sanchoy, and Z. Gungor. 2010. A genetic algorithm based heuristic for scheduling of virtual manufacturing cells (VMCs). *Computers and Operations Research* 37:1148–56. doi:10.1016/j.cor.2009.10.006.

Kessen, S. E., M. D. Toksari, Z. Gungor, and E. Guner. 2009. Analyzing the behaviors of virtual cells (VCs) and traditional manufacturing systems: Ant Colony optimization (ACO)-based meta-models. *Computers and Operations Research* 36 (7):2275–85. doi:10.1016/j.cor.2008.09.002.

Kia, R., A. Baboli, N. Javadian, R. Tavakkoli-Moghaddam, M. Kazemi, and J. Khorrami. 2012. Solving a group layout design model of a dynamic cellular manufacturing system with alternative process routings, lot splitting and flexible reconfiguration by simulated annealing. *Computer and Operations Research* 39:2642–58. doi:10.1016/j.cor.2012.01.012.

Ko, K. C., and P. J. Egbelu. 2000. Performance comparison of static and dynamic cellular manufacturing system. Proceedings of the First Group Technology/Cellular Manufacturing World Symposium, 1–60. San Juan, Puerto Rico.

Ko, K. C., and P. J. Egbelu. 2003. Virtual cell formation. *International Journal of Production Research* 41 (11):2365–89. doi:10.1080/0020754031000087193.

Ma, H., and D. Simon. 2011. Blended biogeography-based optimization for constrained optimization. *Engineering Applications of Artificial Intelligence* 24(3): 517–525.

Mahdavi, I., A. Aalaei, M. M. Paydar, and M. Solimanpur. 2010. Designing a mathematical model for dynamic cellular manufacturing systems considering production planning and worker assignment. *Computer and Mathematics with Applications* 60:1014–25. doi:10.1016/j.camwa.2010.03.044.

Mak, K. L., P. Peng, X. X. Wang, and T. L. Lau. 2009. An ant colony optimization algorithm for scheduling virtual cellular manufacturing systems. *International Journal of Computer Integrated Manufacturing* 20 (60):524–37. doi:10.1080/09511920600596821.

Mansouri, S. A., S. M. Moattar Husseini, and S. T. Newman. 2000. A review of the modern approaches to multi-criteria cell design. *International Journal of Production Research* 38:1201–18. doi:10.1080/002075400189095.

McLean, C. R., H. M. Bloom, and T. H. Hopp. 1982. The virtual manufacturing cell. Proceedings of Fourth IFAC/IFIP Conference on Information Control Problem in Manufacturing Technology, 105–11.

Nomden, G., J. Slomp, and N. C. Suresh. 2006. Virtual manufacturing cells: A taxonomy of past research and identification of future research issues. *International Journal of Flexible Manufacturing Systems* 17:71–92. doi:10.1007/s10696-006-8122-1.

Rajasomashekar, S., and P. Aravindhababu. 2012. Biogeography based optimization technique for best compromise solution of economic emission dispatch. *Swarm and Evolutionary Computation* 7:47–57. doi:10.1016/j.swevo.2012.06.001.

Rheault, M., J. R. Drolet, and G. Abdulnour. 1995. Physically reconfigurable virtual cells: A dynamic model for a highly dynamic environment. *Computers and Industrial Engineering* 29 (1–4):221–25. doi:10.1016/0360-8352(95)00075-C.

Saad, S. M., A. Baykasoglu, and N. N. Z. Gindy. 2002. An integrated framework for reconfiguration of cellualar manufacturing systems using virtual cells. *Production Planning and Control* 13 (4):381–93. doi:10.1080/09537280210130478.

Shahvari, O., N. Salmasi, and R. Logendran. 2009. A meta-heuristic algorithm for flexible flow shop sequence dependent group scheduling problem. Proceedings of the 2009 International conference on Value Chain Sustainability (ICOVACS 2009), Kentucky, USA.

Shahvari, O., N. Salmasi, R. Logendran, and B. Abbasi. 2012. An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems. *International Journal of Production Research* 50:4237–54. doi:10.1080/00207543.2011.604051.

Shahvari, O., and R. Logendran. 2015. Bi-criteria batch scheduling on unrelated-parallel machines. Proceedings of the 2015 Industrial and Systems Engineering Research Conference (ISERC2015), Tennessee, USA.

Shahvari, O., and R. Logendran. 2016a. Bi-criteria batch scheduling in hybrid flow shop. Proceedings of the 2016 Industrial and Systems Engineering Research Conference (ISERC2016), California, USA.

Shahvari, O., and R. Logendran. 2016b. Hybrid flow shop batching and scheduling with a bi-criteria objective. *International Journal of Production Economics* 179:239–58. doi:10.1016/j.ijpe.2016.06.005.

Shahvari, O., and R. Logendran. 2017. An enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes. *Computers and Operations Research* 77:154–76. doi:10.1016/j.cor.2016.07.021.

Simon, D. 2008. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation* 12 (6):702–13. doi:10.1109/TEVC.2008.919004.

Simon, D., A. Shah, and C. Scheidegger. 2013. Distributed learning with biogeography-based optimization: Markov modeling and robot control. *Swarm and Evolutionary Computation* 10:12–24. doi:10.1016/j.swevo.2012.12.003.

Slomp, J., B. V. Chowdary, and N. C. Suresh. 2005. Design of virtual manufacturing cells: A mathematical programming approach. *Robotics and Computer-Integrated Manufacturing* 21:273–88. doi:10.1016/j.rcim.2004.11.001.

Tambuskar, D. P., B. E. Narkhede, and S. S. Mahapatra. 2015. A novel algorithm for virtual cellular manufacturing considering real life production factors. *International Journal of Services and Operations Management*. doi:10.1504/IJSOM.2015.067479.

Vakharia, A. J., J. P. Moily, and Y. Huang. 1999. Evaluating virtual cells and multistage flowshops. *International Journal of Flexible Manufacturing Systems* 11 (3):291–314. doi:10.1023/A:1008117329327.

Wemmerlow, U., and N. L. Hyer. 1989. Cellular manufacturing in the U.S. industry: A survey of users. *International Journal of Production Research* 27 (9):1511–30. doi:10.1080/00207548908942637.

Yang, T., Y. Kuo, C. H. Hsieh, and W. C. Ge. 2016. An exploratory study of virtual cell design for thin-film transistor–Liquid crystal display (TFT-LCD) array manufacturing. *The International Journal of Advanced Manufacturing Technology* 83 (1):633–44. doi:10.1007/s00170-015-7588-y.